Neuronale Netze mit erweiterten bayesschen Methoden für reale Datensammlungen

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Karsten Ernst Weber

aus

Leverkusen

Burscheid 2003

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen-Friedrich-Wilhelms-Universität Bonn

1. Referent: Prof. Dr. Joachim K. Anlauf
2. Referentin: Prof. Dr. Christel Baier

Tag der Promotion: 19. Dezember 2003

Hilfsmittel

An Eides Statt versichere ich, dass ich die vorliegende Arbeit unter Leitung von Herrn Prof. Dr. Joachim K. Anlauf und Frau Prof. Dr. Christel Baier als Koreferentin selbst und ohne jede unerlaubte Hilfe angefertigt habe, dass diese oder eine ähnliche Arbeit noch keiner anderen Stelle zur Prüfung vorgelegen hat und dass sie nur an den angegebenen Stellen auszugsweise veröffentlicht worden ist.

Danksagung

Mein Dank gilt allen Mitarbeitern des Projekts PRINCE für die äußerst konstruktive und menschlich sehr angenehme Zusammenarbeit.

Insbesondere danke ich meinem Doktorvater Prof. Anlauf für die Betreuung und Unterstützung während der Projektzeit, für die Einbringung vieler Ideen und Literaturquellen und ganz besonders für die Vermittlung des Themas. Weiter danke ich Herrn Schlagner für die Überlassung des Themas und Unterstützung von Seiten der Bayer AG. Prof. Gervens und Herrn Schweier danke ich für die Einbringung vieler Fragestellungen und Ideen aus unterschiedlichen Blickwinkeln sowie weiteren Hinweisen auf Literaturquellen.

Allen Diplomanden — Herrn Azizi, Herrn Steinmeier, Frau Vieten und Herrn Wendler — und allen weiteren Softwareentwicklern — Herrn Crone, Herrn Wedemeyer und Frau Wollermann — danke ich für ihre engagierte Mitarbeit im Projekt. Besonderer Dank gilt dabei Herrn Wendler, der darüber hinaus während seines Praxissemesters und seiner Festanstellung für das Projekt entscheidende Arbeit geleistet hat.

Mein Dank gilt auch allen Mitarbeitern der Bayer AG, die für ein technisch funktionierendes Umfeld gesorgt haben, insbesondere Herrn Guntermann, Herrn Körner und Frau Krohn-Huppertz.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Künstliche neuronale Netze sind ein weit verbreiteter Ansatz zur Lösung nicht-linearer Regressions- und Klassifikationsprobleme. Während die klassische Methodik der neuronalen Netze zwar auf der einen Seite gute Lösungen hervorgebracht hat, ist ihr praktischer Einsatz problematisch, wenn eine gute Generalisierungsfähigkeit bei vollautomatischem Training auf unterschiedlichsten Trainingsdatenmengen benötigt wird. Die sogenannten bayesschen Methoden, die durch D. J. C. MacKay 1992 veröffentlicht wurden ([MacKay1], [MacKay2]), besitzen demgegenüber eine Reihe von verbesserten Eigenschaften und eignen sich daher als Basis für eine weitere Entwicklung.

Die vorliegende Arbeit beschreibt eine Reihe von Methoden, die für die Verarbeitung von Korrosionsdaten mit künstlichen neuronalen Netzen und bayesschen Methoden entwickelt und implementiert wurden. Alle Methoden und ihre Beschreibungen sind aber allgemein gehalten, sodass sie auch auf viele andere reale Regressions- und Klassifikationsprobleme angewendet werden können. Zu diesen Problemen gehören insbesondere solche der Technik, der Naturwissenschaften, der Medizin, der Ökonomie oder anderen empirischen Wissenschaften, bei denen Daten durch viele verschiedenartige Messgrößen beschrieben werden.

Der entscheidende Schwerpunkt liegt hier auf der Verarbeitung von realen Daten. Dies umfasst insbesondere Datensammlungen, die ursprünglich nicht zur Verarbeitung durch neuronale Netze angelegt wurden. Bei der Verarbeitung derartiger Daten treten zahlreiche Probleme auf, für die derzeit keine oder nur unzureichende Lösungen bekannt sind.

Häufig stammen die Daten einer realen Datensammlung aus unterschiedlichen Quellen und sind auf unterschiedliche Art beschrieben. Es wird daher eine systematische Methodik entwickelt, mit deren Hilfe die Ursprungsdaten intensiv vorverarbeitet werden können. Diese Vorverarbeitung berücksichtigt dabei Vorwissen über die Herkunft der Daten und über das zugrunde liegende Phänomen und verbessert so entscheidend die Generalisierungsfähigkeit des Gesamtsystems. Für fehlende Werte („missing values“) wird ein spezielles Modell vorgestellt und eine algorithmisch effiziente Lösung erarbeitet.

Liegen den Trainingsdaten verschiedene Messverfahren zugrunde, so muss jedem Datensatz eine eigene Messgenauigkeit zugeordnet werden. Die bayesschen Methoden werden daher so erweitert, dass jeder Datensatz mit einem individuellen Fehler trainiert werden kann. Dies wiederum ermöglicht eine sehr robuste Berechnung von Fehlerangaben der Prognose, die als Prognosekonfidenz zu interpretieren ist. Der Zusammenhang zwischen Trainings- und Prognosefehlern wird intensiv analytisch und empirisch untersucht.

Reale Trainingsdaten können leicht einen extrem großen Eingaberaum und eine starke Clusterung besitzen, wenn sie ursprünglich als reine Datensammlungen angelegt wurden. Daher wird ein Verfahren entwickelt, das auch Eingaberäume dynamischer Dimension verarbeiten und dabei eine Clusterung der Daten berücksichtigen kann. In die Clusterbildung fließt dabei Vorwissen eines Fachmanns über das Problem ein.

Der Zusammenhang zwischen Eingangs- und Ausgangsgrößen ist im Falle der Korrosion nicht überall deterministisch, sondern enthält in bestimmten Regionen Zufallselemente. Das Problem regional unterschiedlichen inhärenten Rauschens wird daher derart gelöst, dass das Rauschen explizit erkannt und in Form einer besonderen Prognosefehlerkomponente angezeigt wird.

Bei der Korrosion ergeben sich diskontinuierliche Ausgangsgrößen funktional aus den Eingängen, das bekannte Klassifikationsmodell der „class conditional density estimation“ ist somit nicht anwendbar. Daher wird ein alternatives Modell zur Klassifikation entwickelt, das die funktionale Abhängigkeit der Wahrscheinlichkeitsverteilung der Ausprägungen der Ausgangsgrößen von den Eingangsgrößen berücksichtigt.

Alle Lösungen zu speziellen Teilproblemen des Phänomens und der Daten werden zu einem Gesamtsystem zusammengefügt, eine Implementierung speziell für die Verarbeitung von Korrosionsdaten liegt vor. Sämtliche verwendeten Algorithmen sind detailliert beschrieben und effizient. Insbesondere das Training der Netze wird vollautomatisch durchgeführt: sowohl die Vorverarbeitung der Daten als auch die Bestimmung aller Netzparameter (Gewichte, Gewichtsregularisierung, Anzahl der Neuronen) wird für unterschiedlichste Trainingsdaten automatisch durchgeführt. Dabei wird kein Testset benötigt, es werden also alle Trainingsdaten voll genutzt.

Selbstverständlich werden verschiedene Komponenten und das Gesamtsystem als Ganzes empirisch untersucht. Die Ergebnisse entsprechen dabei voll den Erwartungen.

Es gibt bereits Veröffentlichungen anderer Autoren über Anwendungen künstlicher neuronaler Netze in der Werkstofftechnik ([BulHoo], [SchBroRee]). Diese verwenden jedoch nur kleine Bereiche und eng abgegrenzte Trainingsdaten, sodass viele der hier diskutierten Probleme gar nicht erst auftreten. Trotzdem sind auch diese Ergebnisse so vielversprechend, dass konkrete Adaptionen neuronaler Netze auf Korrosionsprobleme wertvoll erscheinen.

1.1 Das Projekt PRINCE: Chronologie

Im Vorfeld des Projekts PRINCE (prognosis by intelligent networks for corrosion engineering) wurden von der Bayer AG, Leverkusen, in der Abteilung Werkstofftechnik eine Reihe von Projekten zur elektronischen Erhebung und Speicherung von Korrosionsfakten durchgeführt.

Das erste dieser Projekte war 1988 der Aufbau der KISS-Datenbank (Korrosionsinformationssystem). Diese relationale Datenbank umfasst derzeit etwa 80 000 Datensätze (Korrosionssysteme) unterschiedlicher Quellen, die in einem stark formalisierten und sehr detaillierten Datenschema beschrieben sind. Die KISS-Datenbank enthält neben den Korrosionsfakten selbst auch Felder mit und Verweise zu administrativen und betriebswirtschaftlichen Informationen.

Das KISS-Datenbankschema wurde über die Jahre ergänzt und weiterentwickelt, es entstanden Softwaremodule zur Eingabe, Recherche, Import und Reportgenerierung. Seit 1994 wird der Einsatz von neuronalen Netzen mit Korrosionsdaten erprobt und dabei eine Architektur mit drei Modulen verfolgt: Modul 1 selektiert Korrosionsdaten aus der Datenbank, die Modul 2 dann als Trainingsdaten für einzelne Netze verwendet; die trainierten Netze werden in der Datenbank abgelegt. Modul 3 kann dann zeitlich versetzt mit Hilfe der abgelegten Netze Prognosen zu Korrosionsfragestellungen berechnen. 1999 wurde ein Prototyp für Modul 3 fertig gestellt, der die automatische Auswahl und Kooperation von mehreren abgelegten Netzen ermöglichte [Möbius].

In den Jahren 2000 bis 2002 wurden die Anstrengungen zur Erstellung eines mathematisch fundierten und gleichzeitig praxistauglichen Gesamtsystems zur Prognose von Korrosionsverhalten intensiviert. Kooperationspartner in diesem Zeitraum waren die Bayer Werkstofftechnik (BTS-PT-WT 1 LEV), die EDV der technischen Entwicklung (BTS-BPS IT), die FH Osnabrück und die Universität Bonn. In dieser Zeit entstanden auch die Konzepte der vorliegenden Arbeit, eine Praxissemesterarbeit [Wendler1] sowie insgesamt vier Diplomarbeiten [Steinmeier], [Azizi], [Vieten] und [Wendler2].

1.2 Übersicht über die Arbeit

Kapitel 2 führt kurz in die bayesschen Methoden ein, wie sie ursprünglich von MacKay veröffentlicht wurden. Es beschränkt sich dabei auf diejenigen Teile, die für ein Verständnis der weiteren Kapitel sinnvoll sind, und ergänzt einige wichtige Anmerkungen.

Kapitel 3 stellt den speziellen, in der Implementierung verwendeten Netztyp sowie die zugehörige bayessche Theorie dar. Neben der theoretischen Beschreibung werden ein effizienter Trainingsalgorithmus

und zahlreiche Eigenschaften des Netzverhaltens hergeleitet. Die Darstellung empirischer Auswertungen schließt das Kapitel ab.

In Kapitel 4 werden aufbauend auf Netzen des Kapitels 3 Lösungen für drei spezielle praktische Probleme diskutiert. Diese umfassen die Kooperation von Netzen zur Lösung des Clusterungsproblems und des Problems fehlender Werte, ein zur üblichen Klassifikation alternatives Modell sowie die Erkennung von regionalem Rauschen.

Kapitel 5 führt Methoden eines (korrosions-)problemangepassten Datenmodells ein, das eine inhaltlich saubere und benutzerfreundliche Beschreibung von Korrosionsdaten ermöglicht. Es werden die Abbildung der ursprünglichen KISS-Daten in Daten dieses Schemas und die anschließende Abbildung der Daten dieses Schemas auf Trainings- und Prognosedaten der Netze informell beschrieben, wobei stets die Methodik in den Vordergrund gestellt wird.

Kapitel 6 fasst kurz die wesentlichen Leistungsmerkmale der Softwareimplementierung zusammen.

Empirische Auswertungen zum Gesamtsystem, die die Korrektheit und Leistungsfähigkeit demonstrieren, werden in Kapitel 7 vorgestellt.

Kapitel 2

Grundlagen: bayessche Methoden nach MacKay

Regressions- und Klassifikationsprobleme, zusammenfassend Generalisierungsprobleme genannt, sind häufige praktische Fragestellungen in verschiedensten Wissenschaften und deren Anwendungen. Viele Generalisierungsprobleme lassen sich mit Hilfe von künstlichen neuronalen Netzen lösen, deren Methodik einen wichtigen Teil der Informatik darstellt.

Bei der Anwendung von neuronalen Netzen auf Generalisierungsprobleme unterscheidet man zwischen klassischen und bayesschen Methoden. Bayessche Methoden kennzeichnen sich dadurch, dass sie durchgehend in allen Teilaspekten wahrscheinlichkeitsorientiert sind: alle beschriebenen Größen können Zufallsvariablen sein, in praktischen Implemetierungen sind es die meisten auch. Dies hat enorme Auswirkungen auf den Trainingsprozess und anschließende Prognosen eines Netzes.

Dieses Kapitel beschreibt bayessche Methoden, wie sie in der Literatur diskutiert werden. Es beschränkt sich aber auf die Teile, die für ein Verständnis der weiteren Kapitel wichtig sind, und fügt einige wesentliche Aspekte hinzu. In Abschnitt 3.1 wird dann eine alternative Konkretisierung bayesscher Methoden vorgestellt, auf der die vorliegende Implementierung aufbaut.

Die historisch wichtigsten Quellen zu bayesschen Methoden sind die Originalveröffentlichungen von MacKay ([MacKay1], [MacKay2]) und das leichter verständliche Buch von Bishop ([Bishop]). In diesem Kapitel werden nur Regressionsprobleme angesprochen, die Behandlung von Klassifikationsproblemen wird etwa in [MacKay3] und [Bishop] diskutiert. Weitere interessante Anwendungen und Betrachtungen von bayesschen Methoden finden sich unter anderem in [BioMeePot], [LamVeh], [MacKay4], [MülIns], [PenRob], [SykDorRap], [Thodberg], [WatMacRob], [WilQazBis], [Williams] und [ZhuRoh].

2.1 Die bayessche Gleichung

Bayessche Methoden beurteilen die Generalisierungsfähigkeit eines Modells anhand der Wahrscheinlichkeit, mit der dieses Modell die gegebenen Trainingsdaten erklärt. Für ein gegebenes Modell $\mathcal{H}$ und gegebene Trainingsdaten D gilt die bayessche Gleichung

$$P(\mathcal{H}|D) \quad = \quad \frac{P(\mathcal{H})P(D|\mathcal{H})}{P(D)}. \tag{2.1}$$

Die Größe $P(\mathcal{H}|D)$ auf der linken Seite dieser Gleichung wird a posteriori Wahrscheinlichkeit für das Modell $\mathcal{H}$ genannt und bildet ein Maß für die Generalisierungsfähigkeit des Modells $\mathcal{H}$. Man beachte, dass mit dieser Größe verschiedene Modelle miteinander verglichen werden können. Die Größen auf der rechten Seite werden bei der Implementierung bayesscher Methoden so gewählt, dass sie berechnet werden können:

- $P(\mathcal{H})$ ist die sogenannte a priori Wahrscheinlichkeit für das Modell $\mathcal{H}$. Sie sollte das allgemeine Wissen repräsentieren, das über das zugrunde liegende Phänomen, im vorliegenden Fall die Korrosion, bekannt ist, sie sollte aber nicht von den konkreten Trainingsdaten abhängen. In der Praxis

ist meist nur sehr wenig analytisches Wissen über das zugrunde liegende Phänomen bekannt, daher wird man sogenannte nicht-informative a priori Wahrscheinlichkeiten wählen, siehe dazu [Berger].

- $P(D|\mathcal{H})$ ist die Wahrscheinlichkeit dafür, dass die Trainingsdaten D beobachtet worden wären, falls das Modell $\mathcal{H}$ exakt das zugrunde liegende Phänomen beschreiben würde. Diese Wahrscheinlichkeit basiert sehr wesentlich auf der Modellierung der Messfehler.

- $P(D)$ hängt nicht vom Modell ab. Unter der Annahme, dass mindestens eines der betrachteten Modelle $\mathcal{H}_1, \mathcal{H}_2, \ldots$ das zugrunde liegende Phänomen exakt beschreibt, gilt $P(D) = \sum_i P(\mathcal{H}_i)P(D|\mathcal{H}_i)$. In praktischen Implementierungen begnügt man sich mit einer Menge von Modellen, die das zugrunde liegende Phänomen hinreichend genau approximieren; $P(D)$ dient dann lediglich als Normierungsfaktor für die Wahrscheinlichkeiten.

Die bayessche Gleichung 2.1 beschreibt ganz allgemein die bayesschen Methoden für neuronale Netze. Entscheidend ist nun für jede konkrete Anwendung die Wahl einer Menge von Modellen $\mathcal{H}_1, \mathcal{H}_2, \ldots$ sowie deren a priori Wahrscheinlichkeit $P(\mathcal{H}_i)$ und Trainingsdatenwahrscheinlichkeit $P(D|\mathcal{H}_i)$.

2.2 Neuronale Netze und bayessche Methoden

Historisch gesehen bildeten bayessche Methoden zunächst eine Erweiterung der klassischen neuronalen Netze, was die Form der Trainingsdaten, der Netzfunktion und des Fehlermodells beeinflusste.

Es wird angenommen, dass das zugrunde liegende Phänomen durch eine Funktion $f : \mathbb{R}^L \to \mathbb{R}$ exakt beschrieben werden kann[1]. Die Funktion f, im Folgenden die wahre Funktion genannt, bildet einen Vektor von Eingangsgrößen x deterministisch auf eine Ausgangsgröße, in der Literatur oft Zielgröße genannt, ab. Die wahre Funktion ist natürlich unbekannt und soll bestimmt werden. Während klassische neuronale Netze (meist) versuchen, die wahre Funktion zu schätzen, also eine Approximation zu berechnen, bestimmen bayessche Methoden eine Verteilung von möglichen wahren Funktionen.

Das zugrunde liegende Phänomen wurde nun an N Stellen $x_n \in \mathbb{R}^L$ beobachtet, wobei der beobachtete Wert mit $t_n \in \mathbb{R}$ bezeichnet werden soll. Die Gesamtheit der Trainingsdaten ist daher

$$D \quad = \quad \{(x_1, t_1), \ldots, (x_N, t_N)\}\,. \tag{2.2}$$

Für jeden beobachteten Wert t_n wird angenommen, dass er durch Überlagerung des wahren Funktionswerts an der Stelle der Beobachtung $f(x_n)$ mit einem normalverteilten Rauschen mit Erwartungswert 0 entstanden ist. Dieses Rauschen wird auch Messfehler genannt und soll die Varianz β^{-1} haben[2]. Es gilt

$$t_n \quad \propto \quad \mathcal{N}\left(f(x_n), \beta^{-1}\right), \qquad n = 1, \ldots, N. \tag{2.3}$$

Klassische neuronale Netze werden durch ihre Netzfunktion $g : \mathbb{R}^L \times \mathbb{R}^M \to \mathbb{R}$ beschrieben, die jedem Eingangsvektor $x \in \mathbb{R}^L$ und jedem sogenannten Gewichtsvektor $w \in \mathbb{R}^M$ einen Funktionswert zuordnet. Ziel ist es, einen Gewichtsvektor $\hat{w}$ zu finden, sodass die Funktion $g(., \hat{w})$ die Funktion $f(.)$ gut approximiert.

Bayessche Methoden betrachten dagegen eine Verteilung von Gewichten, die berechnet werden soll. Formal wird hier ein Modell $\mathcal{H}$ mit einem Gewichtsvektor w gleichgesetzt. Für die Wahrscheinlichkeit, den Wert t_n an der Stelle x_n zu beobachten, falls w der wahre Gewichtsvektor ist (d.h. $\forall\, x \in \mathbb{R}^L$: $g(x, w) = f(x)$), gilt

$$p(t_n|w) \quad = \quad \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t_n - g(x_n, w))^2\right). \tag{2.4}$$

Da die Stellen $x_1, \ldots, x_N$ der Beobachtungen fest vorgegeben sind, werden sie in der Notation der bedingten Wahrscheinlichkeitsdichten weggelassen. Nimmt man nun die stochastische Unabhängigkeit der

[1] Es genügt hier den Fall einer Ausgangsvariablen zu beschreiben, die Erweiterung auf mehrere Ausgänge ist einfach.

[2] Die Variable β wird hier als gegeben angenommen, Abschnitt 2.4 behandelt ihre Bestimmung. Die Wahl dieser und weiterer Variablen entspricht der Notation in [Bishop], [MacKay1] und [MacKay2].

Beobachtungen an, ergibt sich die Wahrscheinlichkeitsdichte für das Beobachten aller Trainingsdaten zu

$$\begin{aligned} p(D|w) &= \prod_{n=1}^{N} p(t_n|w) \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left(-\frac{\beta}{2}\sum_{n=1}^{N}(t_n - g(x_n, w))^2\right) \end{aligned} \tag{2.5}$$

Nachdem die Trainingsdatenwahrscheinlichkeit bestimmt wurde, muss nun noch die a priori Wahrscheinlichkeit der Gewichte festgelegt werden. Eine objektive Festlegung ist in den meisten Anwendungsfällen nicht möglich, da dazu analytische Kenntnisse über das zugrunde liegende Phänomen nötig wären[3]. Darüber hinaus hängt eine sinnvolle a priori Verteilung von der Funktionalität der einzelnen Gewichte in der Netzfunktion g ab, siehe dazu Abschnitt 2.6. In der Praxis möchte man einfachen, (in x) glatten, wenig gekrümmten Funktionen $g(., w)$ eine höhere a priori Wahrscheinlichkeit zuordnen als komplizierten Funktionen. Bei vielen Netztypen (generalisierte lineare Netze, Feed-Forward-Netze) kann die „Kompliziertheit“ einer Funktion beschränkt werden, indem man nur betragsmäßig kleine Gewichte zulässt. Daher wählt man als a priori Verteilung eines einzelnen Gewichts eine Normalverteilung mit Erwartungswert 0 und Varianz α^{-1}. Die Variable α wird wie β hier als vorgegeben angenommen. Es folgt

$$\begin{aligned} p(w) &= \prod_{m=1}^{M} p(w_m) \\ &= \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left(-\frac{\alpha}{2}\sum_{m=1}^{M} w_m^2\right), \end{aligned} \tag{2.6}$$

wobei festgelegt wurde, dass die einzelnen Gewichte stochastisch unabhängig sein sollen.

Damit sind nun alle Verteilungen festgelegt, die zu einer praktischen Berechnung der Verteilung der Gewichte nötig sind. Nach der bayesschen Gleichung 2.1 ergibt sich

$$p(w|D) = \frac{1}{p(D)}\left(\frac{\alpha}{2\pi}\right)^{M/2}\left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left(-\frac{\alpha}{2}\sum_{m=1}^{M} w_m^2 - \frac{\beta}{2}\sum_{n=1}^{N}(t_n - g(x_n, w))^2\right). \tag{2.7}$$

Die a posteriori Verteilung der Gewichte spielt bei allen weiteren Aspekten der bayesschen Methoden eine wichtige Rolle und muss daher berechnet werden. In der recht allgemeinen Form nach Gleichung 2.7 ist eine analytische Beschreibung der a posteriori Gewichtsverteilung nicht einfacher als durch eben diese Gleichung möglich. Man begnügt sich daher durch eine Approximation. Es gibt nun zwei grundsätzliche Möglichkeiten, derartige Approximationen zu bestimmen:

- Die Verteilung wird durch eine repräsentative Stichprobe beschrieben. Dies kann etwa durch Monte-Carlo-Methoden und Markov-Ketten geschehen.
- Die Verteilung wird durch eine Normalverteilung approximiert. Diese Beschreibung wird im Folgenden verwendet.

Betrachtet wird dazu die Funktion

$$\begin{aligned} S(w) &:= \frac{\alpha}{2}\sum_{m=1}^{M} w_m^2 + \frac{\beta}{2}\sum_{n=1}^{N}(t_n - g(x_n, w))^2 \\ &= -\ln p(w|D) + const. \end{aligned} \tag{2.8}$$

Die Funktion S hat nun genau dort ein Minimum, wo $p(w|D)$ maximal ist: sucht man also ein globales Minimum von S ist dies äquivalent zur Suche des wahrscheinlichsten Gewichtsvektors w_{MP}. Genau dies praktizieren klassische Methoden. Die Funktion S besteht aus zwei Summanden, die für klassische Methoden einzeln interpretierbar sind. Der linke Summand hängt nur von den Gewichten ab und entspricht der

[3] Daher wird die a priori Wahrscheinlichkeit in der Literatur auch oft subjektive Wahrscheinlichkeit genannt.

klassischen Gewichtsregularisierung mit sogenanntem quadratischem Gewichtsfehlerterm. Ein Vergleich zwischen klassischen und bayesschen Verfahren zur Bestimmung der Gewichtsregularisierung findet sich etwa in [AmaMur]. Der rechte Summand beschreibt den quadratischen Fehler auf den Trainingsdaten. Man beachte, dass diese Art des Fehlers auf den Trainingsdaten direkt mit der Art der Verteilung des Messrauschens korrespondiert.

Die Approximation der a posteriori Verteilung der Gewichte besteht nun in einer Approximation der Funktion S durch ihre Taylor-Reihe bis zum Grad 2 an der Stelle des wahrscheinlichsten Gewichtsvektors w_{MP}:

$$S(w) \quad = \quad S(w_{\mathrm{MP}}) + \frac{1}{2}(w - w_{\mathrm{MP}})^T A (w - w_{\mathrm{MP}}) \tag{2.9}$$

mit der Hesse-Matrix $A = \nabla\nabla S(w_{\mathrm{MP}})$. Die a posteriori Verteilung der Gewichte ist nun approximativ

$$p(w|D) \quad = \quad \frac{\sqrt{\det A}}{(2\pi)^{W/2}} \exp\left(-\frac{1}{2}(w - w_{\mathrm{MP}})^T A (w - w_{\mathrm{MP}})\right), \tag{2.10}$$

und man sieht, dass die Hesse-Matrix A der Funktion S an der Stelle w_{MP} die Inverse der Kovarianzmatrix der Gewichte in der a posteriori Verteilung bildet. Der Normierungsfaktor vor der Exponentialfunktion wurde so gewählt, dass $\int p(w|D)\,dw = 1$ ist. Dies entspricht der Annahme, dass mindestens ein Gewichtsvektor w das zugrunde liegende Phänomen exakt beschreibt; diese Annahme wird in der Praxis zwar nie exakt, aber in der Regel bei vernünftiger Wahl der Netzfunktion g in hinreichender Näherung erfüllt.

Um eine konkrete Beschreibung der a posteriori Verteilung der Gewichte zu erhalten, müssen nun der Vektor $w_{\mathrm{MP}} \in \mathbb{R}^M$ und die Matrix $A \in \mathbb{R}^{M \times M}$ berechnet werden. Den wahrscheinlichsten Gewichtsvektor erhält man durch ein Minimierungsverfahren, das auf die Funktion S angewendet wird, und das auch bei klassischen Netzen verwendet wird. Zusätzlich muss noch die Hesse-Matrix von S an der Stelle w_{MP} berechnet werden, was etwa durch zweifache symbolische Differenzierung erreicht werden kann.

An dieser Stelle soll noch darauf hingewiesen werden, dass es mehrere (globale) Minima w_{MP} geben kann, Abschnitt 2.5 geht darauf näher ein. Untersuchungen bezüglich der Güte der Approximation von S finden sich in [MülIns] und [Thodberg].

2.3 Prognosen

Bisher wurde lediglich beschrieben, wie man eine a posteriori Verteilung von Gewichtsvektoren berechnet. Hier soll nun die erste Anwendung dieser Verteilung beschrieben werden.

Zu einem Anfragepunkt x soll das Netz mit der Netzfunktion $g(x, w)$ und der durch das Training berechneten Verteilung $p(w|D)$ eine Prognose berechnen. Präzise ausgedrückt: welche ist die prognostizierte Verteilung möglicher Messwerte t, wenn an der Stelle x gemessen wird ? Die Antwort gibt eine Faltung über die Gewichte:

$$p(t|x, D) \quad = \quad \int p(t|x, w) p(w|D)\, dw. \tag{2.11}$$

Approximiert man die Netzfunktion an der Stelle w_{MP} in den Gewichten linear, $g(x, w) = g_{\mathrm{MP}} + (w - w_{\mathrm{MP}})^T z$ mit den Abkürzungen $g_{\mathrm{MP}} := g(x, w_{\mathrm{MP}})$ und $z := \nabla_w g(x, w_{\mathrm{MP}})$, so kann die Verteilung der Netzprognosen approximativ berechnet werden.

$$\begin{aligned}
p(t|x, D) &= \int \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t - g(x, w))^2\right) \sqrt{\frac{\det A}{(2\pi)^W}} \exp\left(-\frac{1}{2}(w - w_{\mathrm{MP}})^T A (w - w_{\mathrm{MP}})\right) dw \\
&= \sqrt{\frac{\beta \det A}{(2\pi)^{1+W}}} \int \exp\left(-\frac{\beta}{2}(t - g_{\mathrm{MP}} - (w - w_{\mathrm{MP}})^T z)^2 - \frac{1}{2}(w - w_{\mathrm{MP}})^T A (w - w_{\mathrm{MP}})\right) dw \\
&= \sqrt{\frac{\beta \det A}{(2\pi)^{1+W}}} \int \exp\Bigg(-\frac{1}{2}\Big(\beta(t - g_{\mathrm{MP}})^2 - \beta(t - g_{\mathrm{MP}})(w - w_{\mathrm{MP}})^T z
\end{aligned}$$

$$
\begin{aligned}
& -\beta(t-g_{\mathrm{MP}})z^T(w-w_{\mathrm{MP}}) + \beta(w-w_{\mathrm{MP}})^T zz^T(w-w_{\mathrm{MP}})^T \\
& +(w-w_{\mathrm{MP}})^T A(w-w_{\mathrm{MP}})\Big)\Big)\, dw \\
= \quad & \sqrt{\frac{\beta\det A}{(2\pi)^{1+W}}} \int \exp\Big(-\frac{1}{2}\Big(\beta(t-g_{\mathrm{MP}})^2 + \big(w-w_{\mathrm{MP}}-\beta(t-g_{\mathrm{MP}})(A+\beta zz^T)^{-1}z\big)^T \\
& (A+\beta zz^T)\big(w-w_{\mathrm{MP}}-\beta(t-g_{\mathrm{MP}})(A+\beta zz^T)^{-1}z\big) \\
& -\beta^2(t-g_{\mathrm{MP}})^2 z^T(A+\beta zz^T)^{-1}z\Big)\Big)\, dw
\end{aligned} \tag{2.12}
$$

Das gaußsche Integral ([Bishop], appendix B)

$$
\int \exp\left(-\frac{1}{2}v^T M v\right)\, dv \quad = \quad \sqrt{\frac{(2\pi)^{\dim(v)}}{\det M}} \tag{2.13}
$$

kann nun aufgelöst werden.

$$
\begin{aligned}
p(t|x,D) \quad = \quad & \sqrt{\frac{\beta\det A}{(2\pi)^{1+W}}}\sqrt{\frac{(2\pi)^W}{\det(A+\beta zz^T)}} \exp\Big(-\frac{1}{2}\Big(\beta(t-g_{\mathrm{MP}})^2 \\
& -\beta^2(t-g_{\mathrm{MP}})^2 z^T(A+\beta zz^T)^{-1}z\Big)\Big) \\
= \quad & \sqrt{\frac{\beta\det A}{2\pi\det(A+\beta zz^T)}} \exp\left(-\frac{1}{2}(t-g_{\mathrm{MP}})^2 \frac{(\beta-\beta^2 z^T(A+\beta zz^T)^{-1}z)z^T(I+\beta A^{-1}zz^T)z}{z^T(I+\beta A^{-1}zz^T)z}\right) \\
= \quad & \sqrt{\frac{\beta}{2\pi\det(I+\beta A^{-1}zz^T)}} \exp\Big(-\frac{1}{2}(t-g_{\mathrm{MP}})^2 \\
& \cdot\frac{\beta z^T(I+\beta A^{-1}zz^T)z - \beta^2 z^T(A+\beta zz^T)^{-1}(zz^Tz+\beta zz^TA^{-1}zz^Tz)}{z^T(I+\beta A^{-1}zz^T)z}\Big) \\
= \quad & \sqrt{\frac{\beta}{2\pi(1+\beta z^TA^{-1}z)}} \exp\Big(-\frac{1}{2}(t-g_{\mathrm{MP}})^2 \\
& \cdot\frac{\beta z^T(I+\beta A^{-1}zz^T)z - \beta^2 z^T(A+\beta zz^T)^{-1}(A+\beta zz^T)A^{-1}zz^Tz}{z^T(I+\beta A^{-1}zz^T)z}\Big) \\
= \quad & \sqrt{\frac{1}{2\pi(\beta^{-1}+z^TA^{-1}z)}} \exp\left(-\frac{1}{2}(t-g_{\mathrm{MP}})^2 \frac{\beta z^T(I+\beta A^{-1}zz^T)z - \beta^2 z^TA^{-1}zz^Tz}{z^T(I+\beta A^{-1}zz^T)z}\right) \\
= \quad & \sqrt{\frac{1}{2\pi(\beta^{-1}+z^TA^{-1}z)}} \exp\left(-\frac{1}{2}(t-g_{\mathrm{MP}})^2 \frac{\beta z^Tz}{z^Tz+\beta z^TA^{-1}zz^Tz}\right) \\
= \quad & \frac{1}{\sqrt{2\pi(\beta^{-1}+z^TA^{-1}z)}} \exp\left(-\frac{(t-g_{\mathrm{MP}})^2}{2(\beta^{-1}+z^TA^{-1}z)}\right)
\end{aligned} \tag{2.14}
$$

Die Netzprognose $t|D$ an der Stelle x ist somit eine normalverteilte Zufallsvariable mit dem Erwartungswert g_{MP} und der Varianz $\beta^{-1}+z^TA^{-1}z$. Man spricht in diesem Zusammenhang vom Prognosewert $E[t|D]$, von der Prognosevarianz $VAR[t|D]$ und vom Prognosefehler $\sqrt{VAR[t|D]}$. Der Prognosewert $E[t|D] = g_{\mathrm{MP}} = g(x, w_{\mathrm{MP}})$, gleichzeitig der wahrscheinlichste Wert für t, ist keine Überraschung: es ist genau der Wert, den auch klassische Methoden berechnen, indem sie den beim Training gefundenen optimalen Gewichtsvektor w_{MP} in die Netzfunktion einsetzen.

Neben dem Prognosewert ergibt sich aber auch die Prognosevarianz

$$
VAR[t|D] \quad = \quad \beta^{-1} + (\nabla_w g(x, w_{\mathrm{MP}}))^T A^{-1}(\nabla_w g(x, w_{\mathrm{MP}})) \tag{2.15}
$$

in numerisch rechenbarer Form ganz natürlich im bayesschen Kontext. Sie besteht aus zwei Summanden, die zwei voneinander unabhängige Komponenten des Fehlers charakterisieren:

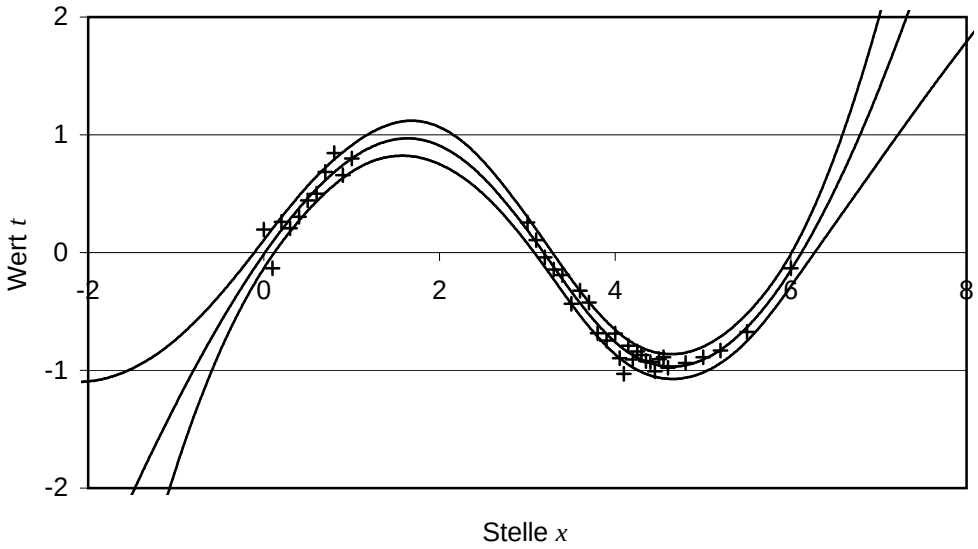

Abbildung 2.1: Beispiel für Prognosen eines Netzes mit bayesschen Methoden. Es wurden 39 Messungen an unterschiedlichen Stellen aus dem Intervall $[0;6]$ mit einem Rauschen (Messfehler) von $\beta^{-1/2} = 0,1$ über der wahren Funktion $f(x) = sin(x)$ simuliert und mit ihnen ein neuronales Netz trainiert. Die Messungen sind als Kreuze dargestellt, die Netzprognose in Form von drei Kurven: die mittlere Kurve stellt den Prognosewert $E[t|D]$ und die beiden äußeren Kurven das einfache Konfidenzintervall dar: $E[t|D] \pm \sqrt{VAR[t|D]}$.

- Der Term β^{-1} stellt die Unsicherheit durch den Messvorgang an der Anfragestelle x selbst dar. Nach Voraussetzung unterliegen alle Messungen einem Messfehlerrauschen mit genau dieser Varianz.

- Der Term $z^T A^{-1} z$ kennzeichnet die Wirkung der Unbestimmtheit der Gewichte auf das Ergebnis der Netzfunktion. Da die Matrix A eine Hesse-Matrix ist, ist sie positiv definit[4] und es gilt $z^T A^{-1} z > 0$. Eine tiefergehende Interpretation des Terms ist schwierig und wenig intuitiv.

Abbildung 2.1 zeigt einige Trainingsdaten zusammen mit der Prognose eines auf diesen Daten trainierten Netzes. Man sieht, dass der Fehler dort, wo die Trainingsdaten dicht nebeneinander liegen, der Prognosefehler $\sqrt{VAR[t|D]}$ etwa gleich dem Messfehler $\beta^{-1/2}$ ist, abseits der Messdaten aber stark ansteigt. Der Prognosefehler ist also auch ein Maß für die Dichte der Trainingsdaten.

An dieser Stelle ist zu bemerken, dass der Prognosefehler sehr wesentlich von der Menge der zur Verfügung gestellten Funktionen $\{g(.,w)|p(w) > 0\}$ und von deren a priori Wahrscheinlichkeiten abhängt. Ist die Menge der möglichen Funktionen zu klein gewählt, dann hat das Netz auch nach dem Training wenig Spielraum für alternative Gewichte, was bedeutet, dass der Prognosefehler klein sein wird. In dieser Situation kann von einem kleinen Prognosefehler nicht mehr auf viel Wissen in Form einer hohen Dichte von Trainingsdaten geschlossen werden.

In extremen Fällen kann der Prognosefehler auch völlig unbedeutend werden. Betrachtet man etwa die Funktion $g(x,w) := x^T w$ mit $x, w \in \mathbb{R}^L$, dann ist für den Anfragepunkt $x = 0$ und jedes beliebige Gewicht w die Netzausgabe $g(0,w) = 0$. Anschaulich betrachtet heißt dies: an der Stelle 0 ist sich das Netz völlig sicher, dass 0 der wahre Funktionswert ist. Es kommt sogar immer zu dieser Aussage, also unabhängig von der a priori Verteilung der Gewichte oder den Trainingsdaten.

Es ist also wichtig, die Netzfunktion $g(x,w)$ und auch die a priori Verteilung der Gewichte $p(w)$ sinnvoll zu wählen. Zwischen beiden besteht eine enge inhaltliche Verbindung, die in Abschnitt 2.7 näher beschrieben wird.

[4] Als Hesse-Matrix ist sie zunächst nur positiv semidefinit. Die echt positive Definitheit erhält sie durch die Gewichtsregularisierung: linker Summand der Gleichung 2.8.

2.4 Bestimmung der Hyperparameter α und β

In Abschnitt 2.2 wurden die Variablen α, das die a priori Verteilung der Gewichte steuert, und β, das den Messfehler der Daten beschreibt, als bekannt vorausgesetzt. Dies ist in der Praxis meist nicht der Fall. Führt man den Gedanken der bayesschen Methoden an dieser Stelle nun konsequent fort, so müssen α und β als Zufallsvariablen beschrieben werden. Da ein Training aber wie in Abschnitt 2.2 beschrieben auch für feste, a priori gewählte Größen α und β möglich ist, werden diese Variablen auch Hyperparameter genannt.

Auch für Hyperparameter gilt die bayessche Gleichung 2.1

$$p(w, \alpha, \beta|D) \quad = \quad \frac{p(w, \alpha, \beta)p(D|w, \alpha, \beta)}{p(D)}. \tag{2.16}$$

Die Wahrscheinlichkeiten auf der rechten Seite der Gleichung müssen nun bestimmt werden. Zunächst werden die einzelnen Wahrscheinlichkeiten auf ihre Abhängigkeiten hin untersucht: β soll a priori unabhängig von w und α sein, also gilt $p(w, \alpha, \beta) = p(w, \alpha)p(\beta)$. Nach Gleichung 2.6 ist zwar die Abhängigkeit der w-Verteilung von α gegeben, α selbst soll aber unabhängig von w beschrieben werden und es gilt $p(w, \alpha) = p(w|\alpha)p(\alpha)$. Weiter ist die Datenwahrscheinlichkeit nach Gleichung 2.5 nicht von der a priori Verteilung der Gewichte, wohl aber vom Messfehler abhängig und es gilt daher $p(D|w, \alpha, \beta) = p(D|w, \beta)$. Dies ergibt zusammen die Form

$$p(w, \alpha, \beta|D) \quad = \quad \frac{p(w|\alpha)p(\alpha)p(\beta)p(D|w, \beta)}{p(D)}. \tag{2.17}$$

Die Wahrscheinlichkeiten $p(D|w, \beta)$ und $p(w|\alpha)$ sind bereits durch die Gleichungen 2.5 und 2.6 gegeben. Um a priori Wahrscheinlichkeiten für α und β festzulegen, stellt MacKay folgende Überlegungen an (siehe dazu auch [Berger]): sowohl $\alpha \in \mathbb{R}^+$ als auch $\beta \in \mathbb{R}^+$ stellen sogenannte Skalierungsgrößen dar, jede Größenordnung ist möglich und soll gleiche Wahrscheinlichkeit erhalten. Aus dieser Überlegung folgen $p(\ln \alpha) = p(\ln \beta) = const$, also

$$p(\alpha) \; = \; \frac{const}{\alpha} \qquad \text{und} \qquad p(\beta) \; = \; \frac{const}{\beta}. \tag{2.18}$$

Diese Wahrscheinlichkeitsdichten sind nicht normalisierbar (engl. improper), d.h. es gibt keine Konstante $const$, sodass $\int p(\alpha)\, d\alpha = 1$ ist. Dies stellt aber kein Problem dar, wenn die a posteriori Wahrscheinlichkeiten für α und β wieder normalisierbar sind.

Mit diesen Festlegungen ist nun die Erweiterung des Modells um zu bestimmende Hyperparameter α und β abgeschlossen. Gleichung 2.11 zur Berechnung von Prognosen lautet nun

$$p(t|x, D) \quad = \quad \int\int\int p(t|x, w, \beta)p(w, \alpha, \beta|D)\, dw\, d\alpha\, d\beta \tag{2.19}$$

Für eine praktische Realisierung derartiger Prognosen schlägt [Bishop] nun zwei Lösungen vor. Eine Möglichkeit ist, analog zu Abschnitt 2.2 die wahrscheinlichsten Werte w_{MP}, α_{MP} und β_{MP} der Verteilung nach Gleichung 2.17 durch ein iteratives Verfahren, welches abwechselnd w einerseits und α und β andererseits optimiert, zu berechnen[5]. Um die Beschreibung der Verteilung des Tripels (w, α, β) einfach zu halten, wird angenommen, dass die Hyperparameter sehr scharf bestimmt sind, also eine sehr geringe a posteriori Varianz besitzen. Daher erhalten α und β als Ergebnis des Trainings die scharfen Werte α_{MP} bzw. β_{MP}. Für die Prognose selbst wird α ohnehin nicht benötigt, und β taucht lediglich im Erwartungswert beim Prognosefehler auf.

[Bishop] schlägt für praktische Implementierungen dieses Verfahren vor. Allerdings ist alternativ auch eine teilweise analytische Erschließung der Verteilung möglich: die Variable α kann nämlich durch Integration eliminiert werden. Setzt man Gleichung 2.17 in Gleichung 2.19 ein, so erhält man die folgende

[5] Genau genommen basiert das von Bishop vorgeschlagene Verfahren namens „evidence approximation“ nicht auf den Gleichungen 2.17 oder 2.19, sondern auf der Maximierung der Ausdrucks $p(D|\alpha, \beta)$ in α und β. Der wesentliche Unterschied dabei ist, dass zur Bestimmung von $p(D|\alpha, \beta)$ keine Wahl von a priori Verteilungen für die Hyperparameter nötig ist.

Form für die Prognoseverteilung:

$$\begin{aligned} p(t|x,D) &= \int\int\int p(t|x,w,\beta)\frac{p(w|\alpha)p(\alpha)p(\beta)p(D|w,\beta)}{p(D)}\,dw\,d\alpha\,d\beta \\ &= \int\int p(t|x,w,\beta)\frac{\left(\int p(w|\alpha)p(\alpha)\,d\alpha\right)p(\beta)p(D|w,\beta)}{p(D)}\,dw\,d\beta. \end{aligned} \tag{2.20}$$

Der Ausdruck $p(w) = \int p(w|\alpha)p(\alpha)\,d\alpha$ beschreibt nun eine neue, erweiterte a priori Verteilung der Gewichte, in der α nicht mehr auftritt. Er kann analytisch berechnet werden:

$$\begin{aligned} p(w) &= \int_0^\infty p(w|\alpha)p(\alpha)\,d\alpha \\ &= \int_0^\infty \left(\frac{\alpha}{2\pi}\right)^{M/2}\exp\left(-\frac{\alpha}{2}\sum_{m=1}^M w_m^2\right)\frac{1}{\alpha}\,d\alpha \\ &= \frac{1}{(2\pi)^{M/2}}\int_0^\infty \alpha^{M/2-1}\exp\left(-\alpha\frac{1}{2}\sum_{m=1}^M w_m^2\right)\,d\alpha \\ &= \frac{1}{(2\pi)^{M/2}}\int_0^\infty \left(\frac{\alpha}{\frac{1}{2}\sum_{m=1}^M w_m^2}\right)^{M/2-1}\exp(-\alpha)\frac{d\alpha}{\frac{1}{2}\sum_{m=1}^M w_m^2} \\ &= \frac{1}{(\pi\sum_{m=1}^M w_m^2)^{M/2}}\int_0^\infty \alpha^{M/2-1}\exp(-\alpha)\,d\alpha \\ &= \frac{\Gamma(M/2)}{(\pi\sum_{m=1}^M w_m^2)^{M/2}}. \end{aligned} \tag{2.21}$$

Dieses Ergebnis kann verwendet werden, um den Trainingsprozess zu vereinfachen und zu beschleunigen, da eine Variable weniger zu bestimmen ist. Formt man Gleichung 2.20 geeignet um, so kann auch β unter Annahme einer scharf bestimmten a posteriori Verteilung durch Integration eliminiert werden, man vereinfacht dadurch den Trainingsprozess weiter. Allerdings ist im Gegensatz zu α eine nachträgliche Bestimmung von β nötig, um Prognosen berechnen zu können, weshalb hier davon abgesehen wird.

Die neue a posteriori Verteilung der Parameter w und β ist nun

$$\begin{aligned} p(w,\beta|D) &= \frac{p(w)p(\beta)p(D|w,\beta)}{p(D)} \\ &= \frac{1}{p(D)}\cdot\frac{\Gamma(M/2)}{\left(\pi\sum_{m=1}^M w_m^2\right)^{M/2}}\cdot\frac{const}{\beta}\left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left(-\frac{\beta}{2}\sum_{n=1}^N(t_n-g(x_n,w))^2\right) \\ &= const\cdot\frac{1}{\left(\sum_{m=1}^M w_m^2\right)^{M/2}\cdot\beta}\beta^{N/2}\exp\left(-\frac{\beta}{2}\sum_{n=1}^N(t_n-g(x_n,w))^2\right) \end{aligned} \tag{2.22}$$

mit der entsprechenden Fehlerfunktion

$$\begin{aligned} S(w,\beta) &= \frac{M}{2}\ln\sum_{m=1}^M w_m^2 + \ln\beta - \frac{N}{2}\ln\beta + \frac{\beta}{2}\sum_{n=1}^N(t_n-g(x_n,w))^2 \\ &= -\ln p(w,\beta|D) + const. \end{aligned} \tag{2.23}$$

Versucht man hier, das wahrscheinlichste Paar $(w_{\mathrm{MP}},\beta_{\mathrm{MP}})$ durch Minimieren der Funktion $S(w,\beta)$ zu bestimmen, stellt man fest, dass die Funktion S nach unten unbeschränkt ist: für $||w|| \to 0$ fällt der erste Summand unbeschränkt, während alle anderen Summanden gegen eine Konstante konvergieren. Die Ursache für dieses Verhalten liegt in der Wahl der a priori Verteilung von α. Diese war zunächst vernünftig, also problembezogen, gewählt. Jedoch lässt sie zu viele kleine Gewichte mit zu großer Wahrscheinlichkeitsdichte zu.

Eine Lösung des Problems soll an dieser Stelle nicht präsentiert werden. Es sei jedoch auf die Abschnitte 3.2.1 und 3.2.2 verwiesen, die eine Lösung für die in der Implementierung verwendeten Netze beschreiben.

2.5 Weitere Hyperparameter und die Evidenz

Die Variablen α und β sind nur zwei Beispiele für Hyperparameter. Es gibt aber viele weitere Parameter, die Modelle $\mathcal{H}$ charakterisieren können:

- die Anzahl der Gewichte M, etwa bestimmt durch die Anzahl innerer Knoten bei feed-forward Netzen, oder die Anzahl von Basisfunktionen bei generalisierten linearen Netzen,
- die Anzahl von Layern bei feed-forward Netzen,
- die Wahl der Aktivierungs- bzw. Basisfunktionen,
- die Menge von möglichen Netzfunktionen g, die in einem Komitee zusammenwirken, ...

Diese Hyperparameter (mit Ausnahme von β) bestimmen die Komplexität eines Netzes, von der wiederum die Generalisierungsfähigkeit abhängt. Bei klassischen Netzen besteht dabei stets die Gefahr des Over- oder Underfittings und es gibt zahlreiche Ansätze diese Gefahren zu vermeiden ([AmaMur], [CibSouGal], [Ripley], [Sarle]), die aber alle gewisse Nachteile aufweisen. Bei bayesschen Methoden dagegen ergeben sich alle komplexitätsbestimmenden Parameter auf natürliche Weise.

Sehr ähnlich wie die aufgeführten expliziten Hyperparameter verhalten sich auch einige technische Parameter, wie etwa die Menge multipler Minima w_{MP} der Funktion $S(w)$, auf die hier auch kurz eingegangen werden soll. Meist ist die a posteriori Verteilung der Gewichte nicht gut durch eine einzelne Normalverteilung zu beschreiben und man versucht daher eine Beschreibung durch einen sogenannten Mix von Normalverteilungen. Gleichung 2.10 wird dann durch die folgende Gleichung ersetzt:

$$p(w|D) \quad = \quad \sum_i P(i) \frac{\sqrt{\det A^{(i)}}}{(2\pi)^{W/2}} \exp\left(-\frac{1}{2}(w - w_{\mathrm{MP}}^{(i)})^T A^{(i)} (w - w_{\mathrm{MP}}^{(i)})\right), \tag{2.24}$$

wobei $P(i)$ die Wahrscheinlichkeit für die i-te Normalverteilung $\mathcal{N}\left(w_{\mathrm{MP}}^{(i)}, A^{(i)}\right)$ darstellt. In Implementierungen unterscheidet man dann zwischen äquivalenten und nicht äquivalenten Minima $w_{\mathrm{MP}}^{(i)}$ der Funktion $S(w)$: zwei gefundene lokale Minima sind äquivalent, wenn sie durch eine Symmetrieoperation in den Gewichten bezüglich der Netzfunktion g und der a priori Verteilung der Gewichte auseinander hervorgehen. Beispiele für äquivalente Minima bei zweistufigen feed-forward Netzen sind die Permutation von Neuronen der verdeckten Schicht samt ihrer Gewichte oder die Vorzeicheninversion aller Gewichte vor und hinter einem verdeckten Neuron, wenn dessen Aktivierungsfunktion symmetrisch zum Ursprung ist. Äquivalente Minima sollten in Gleichung 2.24 analytisch zusammengefasst werden. Nicht äquivalente Minima können danach wie Mitglieder eines Komitees aufgefasst werden.

Betrachten wir hier beispielhaft eine Menge von Modellen $\mathcal{H}_1, \mathcal{H}_2, \ldots$, die in einem Komitee zusammenwirken sollen. Da unbekannt ist, welches dieser Modelle das zugrunde liegende Phänomen korrekt beschreibt, wird nach Bayes eine Wahrscheinlichkeitsverteilung über den einzelnen Modellen beschrieben. Es gilt die bayessche Gleichung

$$P(\mathcal{H}_i|D) \quad = \quad \frac{P(\mathcal{H}_i) p(D|\mathcal{H}_i)}{p(D)}. \tag{2.25}$$

Da die Komiteemitglieder in der Regel gleichrangig sind, werden ihre a priori Wahrscheinlichkeiten gleich sein: $P(\mathcal{H}_1) = P(\mathcal{H}_2) = \ldots$. Daher ist die a posteriori Wahrscheinlichkeit eines Modells $P(\mathcal{H}_i|D)$ direkt proportional zur Wahrscheinlichkeitsdichte, mit der die beobachteten Trainingsdaten durch das Modell $p(D|\mathcal{H}_i)$ erklärt werden. Der Ausdruck $p(D|\mathcal{H}_i)$ spielt daher eine entscheidende Rolle und wird Evidenz des Modells $\mathcal{H}_i$ genannt.

Analog zu Gleichung 2.11 berechnet sich die Prognose eines Komitees als die Verteilung

$$\begin{aligned} p(t|x,D) &= \sum_i p(t|x,\mathcal{H}_i)P(\mathcal{H}_i|D) \\ &= \frac{\sum_i p(t|x,\mathcal{H}_i)p(D|\mathcal{H}_i)}{\sum_i p(D|\mathcal{H}_i)}. \end{aligned} \tag{2.26}$$

Man sieht, dass die Evidenz hier als Gewichtungsfaktor der Prognosen der Komiteemitglieder Verwendung findet. Für das in Abschnitt 2.2 beschriebene Modell gilt für die Evidenz im Rahmen der dortigen Approximation

$$\begin{aligned} p(D) &= \int p(D|w)p(w)\,dw \\ &= \int \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left(-\frac{\beta}{2}\sum_{n=1}^{N}(t_n - g(x_n,w))^2\right) \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left(-\frac{\alpha}{2}\sum_{m=1}^{M} w_m^2\right) dw \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \left(\frac{\alpha}{2\pi}\right)^{M/2} \int \exp\left(-S(w)\right)\, dw \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \left(\frac{\alpha}{2\pi}\right)^{M/2} \int \exp\left(-S(w_{\mathrm{MP}}) - \frac{1}{2}(w-w_{\mathrm{MP}})^T A (w-w_{\mathrm{MP}})\right) dw \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp(-S(w_{\mathrm{MP}})) \frac{(2\pi)^{M/2}}{\sqrt{\det A}} \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \frac{\alpha^{M/2}}{\sqrt{\det A}} \exp\left(-\frac{\alpha}{2}\sum_{m=1}^{M}(w_{\mathrm{MP}})_m^2 - \frac{\beta}{2}\sum_{n=1}^{N}(t_n - g(x_n,w_{\mathrm{MP}}))^2\right). \end{aligned} \tag{2.27}$$

Üblicherweise berechnet man den Logarithmus der Evidenz, da die Evidenz selbst meist nicht mehr mit Gleitkommazahlen darstellbar ist.

$$\begin{aligned} \ln p(D) &= -\frac{\alpha}{2}\sum_{m=1}^{M}(w_{\mathrm{MP}})_m^2 - \frac{\beta}{2}\sum_{n=1}^{N}(t_n - g(x_n,w_{\mathrm{MP}}))^2 \\ &\quad -\frac{1}{2}\ln\det A + \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) \end{aligned} \tag{2.28}$$

Da in Abschnitt 2.2 die Parameter α und β sowie die Dimension des Gewichtsraums M und die Anzahl der Daten N als gegeben vorausgesetzt waren, können die unteren vier Summanden in Implementierungen auch weggelassen werden, wenn alle Komiteemitglieder für diese Größen gleiche Werte besitzen.

In der Praxis wird es oft passieren, dass sich die Evidenzen der einzelnen Modelle um mehrere Größenordnungen unterscheiden; die Unterschiede werden umso größer, je mehr Trainingsdaten und je mehr Gewichte verwendet werden. Der Einfluss der Modelle mit geringerer Evidenz auf die Gesamtprognose ist dann so gering, dass viele Implementierungen schon beim Training diese Modelle ganz aus dem Komitee entfernen. Manche Implementierungen verwenden einfach das Modell mit der höchsten Evidenz. Dies ähnelt dann wieder den klassischen Verfahren, die ein Optimum der Fehlerfunktion über verschiedene Modelle suchen.

Man beachte, dass die Berechnung der Evidenz den Vergleich völlig verschiedener Modelle ermöglicht. Sie ist insbesondere unabhängig von der Dimension des Gewichtsraums oder der Netzstruktur. Durch sie können auch ein Modell mit festen Werten für α und β und eines mit verteilten Größen α und β zusammen in einem Komitee wirken (obwohl diese Konstellation keine sinnvolle a priori Zusammenstellung von Modellen ist).

Nicht mehr vergleichbar sind allerdings Modelle, die unterschiedliche Mengen von inneren Parametern besitzen, denen nicht normierbare a priori Verteilungen zugrunde liegen. Dies soll hier am Beispiel des Hyperparameters β verdeutlicht werden. Aufgrund der Beobachtung, dass β ein Skalierungsparameter ist, wurde eine a priori Verteilung nach Gleichung 2.18 gewählt. Es folgt für die Evidenz eines solchen

Modells

$$\begin{aligned} p(D) &= \int p(D|\beta)\frac{const}{\beta}\,d\beta \\ &= const\int p(D|\beta)\frac{1}{\beta}\,d\beta. \end{aligned} \tag{2.29}$$

Man sieht, dass die Evidenz direkt von der gewählten Konstanten abhängt.

Anders verhält es sich, wenn die Verteilung von β normierbar wird. Nimmt man aufgrund des zugrunde liegenden Phänomens an, dass die Trainingswerte in einer bestimmten Größenordnung liegen und schließt dann zurück auf die mögliche Größenordnung des Messfehlers, so können Schranken für β angegeben werden. Für eine Konstante $\gamma > 0$ seien diese Schranken durch $\exp(-\gamma)$ und $\exp(\gamma)$ gegeben, dann gilt für die neue a priori Verteilung:

$$p(\beta) = \begin{cases} 1/(2\gamma\beta) & : \quad \exp(-\gamma) \le \beta \le \exp(\gamma) \\ 0 & : \quad \text{sonst} \end{cases}. \tag{2.30}$$

Diese a priori Verteilung ist normiert und für die Evidenz gilt

$$p(D) = \frac{1}{2\gamma}\int_{\exp(-\gamma)}^{\exp(\gamma)} p(D|\beta)\frac{1}{\beta}\,d\beta. \tag{2.31}$$

Solange die Evidenzen $p(D|\beta)$ für gegebene β außerhalb des Intervalls $[\exp(-\gamma), \exp(\gamma)]$ vernachlässigbar sind, ist diese Beschränkung von β sinnvoll. Dann (aber nur dann) gilt: je stärker die Einschränkung ist, je kleiner also γ ist, desto größer wird die Evidenz $p(D)$. Man sieht hier ein weiteres Mal, dass die Wahl zwischen einem spezialisierten Modell (γ klein) und einem allgemeinen, flexiblen Modell (γ groß) nicht a priori getroffen werden kann, es sei denn man nutzt Wissen über das zugrunde liegende Phänomen.

Die Auswahl zwischen den verschiedenen Einzelmodellen $\mathcal{H}_1, \mathcal{H}_2, \ldots$ beim Komitee ist ein Hyperparameter. Das Komitee ist somit wieder ein Modell $\mathcal{H}$, dessen Evidenz natürlich berechnet werden kann:

$$p(D|\mathcal{H}) = \sum_i p(D|\mathcal{H}_i)P(\mathcal{H}_i). \tag{2.32}$$

Dieses Prinzip des Erweiterns fester Modelle durch Hyperparameter kann verallgemeinert werden: man spricht dann von sogenannten hierarchischen Modellen. Hierarchische Modelle sind nicht nur in der theoretischen Beschreibung hilfreich, sondern auch bei der Implementierung. Algorithmisch könnte man etwa in einer innersten Schleife die Verteilung der Gewichte berechnen, in einer umschließenden Schleife Verteilungen von α und β bestimmen, und in einer äußeren Schleife ein Komitee über verschiedene Anzahlen von Gewichten bilden.

2.6 Wichtige Erweiterungen

In den Abschnitten 2.2 bis 2.5 wurden konkrete Vorschläge für alle Komponenten, die bei bayesschen Methoden bestimmt werden müssen, gemacht: Netzfunktionen, das Fehlermodell, a priori Verteilungen und Hyperparameter. In diesem Abschnitt werden an verschiedenen Komponenten Veränderungen vorgeschlagen um Modelle zu generieren, die bestimmte Trainingsdaten besser beschreiben können. Während die bisher vorgestellten Komponenten einerseits relativ universell sind und andererseits zu effizienten Implementierungen führen, gelten diese Eigenschaften teilweise für die nachfolgend aufgeführten Erweiterungen nicht mehr.

Die Aufzählung in diesem Abschnitt ist natürlich nicht vollständig. Sie soll vielmehr zeigen, dass bayessche Methoden sehr individuell auf bestimmte Probleme adaptiert werden können, indem analytisches Wissen des zugrunde liegenden Phänomens genutzt wird. Dieses analytische Wissen fließt dann in sehr natürlicher Art in die Gestaltung der Komponenten des Modells ein. Die wesentlichen praktischen Probleme bestehen meist eher darin, derartiges analytisches Wissen zu erlangen.

2.6.1 A priori Verteilungen der Gewichte in Abhängigkeit ihrer Funktionalität

Bei den meisten Netzfunktionen können die Gewichte anhand ihrer Funktionalität gruppiert werden. So bilden etwa bei mehrstufigen Feed-forward-Netzen die Gewichte jeder einzelnen Schicht eine solche Gruppe. Außerdem kann oft zwischen Verbindungs- und Biasgewichten unterschieden werden.

Von Gewichten unterschiedlicher Funktionalität wird unterschiedliches Verhalten verlangt. Gewichte, die als Linearfaktoren zu Eingangsvariablen fungieren, stellen sich in ihrer Größenordnung entsprechend auf die Größenordnung der Eingangsvariablen ein. Im Gegensatz dazu sind Biasgewichte nicht abhängig von der Größenordnung der Eingangsvariablen.

Diese Beobachtung rechtfertigt unterschiedliche a priori Verteilungen der einzelnen Gewichte. Seien $\mathcal{W}_1$ und $\mathcal{W}_2$ zwei Gruppen von Gewichten, dann könnte die a priori Verteilung der Gewichte durch zwei unabhängige Konstanten α_1 und α_2 beschrieben werden:

$$p(w) = \left(\frac{\alpha_1}{2\pi}\right)^{|\mathcal{W}_1|/2} \left(\frac{\alpha_2}{2\pi}\right)^{|\mathcal{W}_2|/2} \exp\left(-\frac{\alpha_1}{2}\sum_{w_m\in\mathcal{W}_1} w_m^2 - \frac{\alpha_2}{2}\sum_{w_m\in\mathcal{W}_2} w_m^2\right). \tag{2.33}$$

Natürlich sind auch mehr als zwei Gruppen denkbar.

Obwohl diese Modellierung möglicherweise der Modellierung des Problems angemessen ist, gibt sie zusätzliche Flexibilität, die durch zusätzliche Hyperparameter Ausdruck findet. Diese zusätzlichen Hyperparameter müssen aber auch wieder bestimmt werden: entweder streng nach Bayes durch eine Verteilung oder heuristisch/approximativ durch die Maximierung der Evidenz. Im Extremfall würde man für jedes einzelne Gewicht ein eigenes α_i bestimmen, was dann α_i als Hyperparameter ad absurdum führen würde.

In der Praxis kann man eine umständliche Einteilung in Gruppen gelegentlich vermeiden, indem man die Eingänge, die Aktivierungsfunktion(en) und die Ausgänge so skaliert, dass alle Gewichte a priori wieder identische Verteilungen besitzen.

2.6.2 Automatic relevance determination

Eine spezielle Art dieser Unterscheidung von Gewichten durch ihre a priori Verteilung wird recht häufig in der Literatur diskutiert und auch praktisch eingesetzt: automatic relevance determination ([BioMeePot], [MacKay4], [PenRob], [Thodberg]). Dieses Verfahren wird bei zweistufigen Feed-forward-Netzen verwendet und fasst jeweils diejenigen Gewichte zu einer Gruppe zusammen, die in der ersten Schicht mit einem bestimmten Eingang verknüpft sind.

Ziel ist es, wichtigen Eingängen einen größeren Einfluss auf die Netzausgänge zu ermöglichen als weniger wichtigen Eingängen. Die Gewichte, die mit den weniger wichtigeren Eingängen verknüpft sind, sollen eine geringere a priori Varianz erhalten. Automatic relevance determination ist somit eine kontinuierliche Variante der bei klassischen Netzen oft diskutierten Feature-Selektion ([Battiti], [Bidasaria], [BleOba], [Kulikowski]).

Die Bestimmung der Hyperparameter $\alpha_1, \ldots \alpha_L$, die mit den Eingängen $x_1, \ldots, x_L$ assoziiert sind, ist problematisch. In der Regel kennt man weder die Reihenfolge der Wichtigkeit der Eingänge noch kann man sie in Form der Hyperparameter quantifizieren. In der Literatur werden verschiedene a priori Verteilungen des Tupels $(\alpha_1, \ldots \alpha_L)$ diskutiert.

2.6.3 Hybridmodelle

Die Wahl der Netzfunktion $g(x, w)$ muss nicht notwendigerweise den üblichen klassischen Netzfunktionen folgen. Insbesondere ist es möglich und sinnvoll Netzfunktionen zu verwenden, die analytisches Wissen über das zugrunde liegende Phänomen nachbilden. Man spricht hier u.a. von Hybridmodellen ([MrzLoo]).

Hybridmodelle bestehen aus sogenannten white boxes und black boxes, die untereinander in Form eines gerichteten zyklenfreien Graphen vernetzt sind. Eine Kante des Graphen besteht aus einer oder mehreren Variablen, für die aber keine Trainingswerte bekannt sein müssen. White boxes sind analytisch bekannte mathematische Funktionen, die keine Parameter (Gewichte) enthalten. Black boxes repräsentieren dagegen unbekannte mathematische Funktionen und werden üblicherweise durch klassische Netzstrukturen mit vielen Gewichten modelliert.

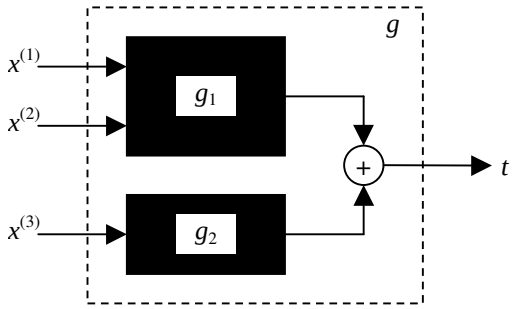

Abbildung 2.2: Beispiel für ein Hybridmodell. Dargestellt ist ein Modell mit den drei Eingängen $x^{(1)}, x^{(2)}$ und $x^{(3)}$ und einem Ausgang t, das aus zwei black boxes g_1 und g_2 und einer white box in Form einer Addition besteht. Es entspricht der Netzfunktion $g(x, w) = g_1(x^{(1)}, x^{(2)}, w_1, \ldots, w_{M'}) + g_2(x^{(3)}, w_{M'+1}, \ldots, w_M)$, wobei die Funktionen g_1 und g_2 nicht näher spezifiziert sind.

Hybridmodelle bilden ein komplexes Thema, im Zusammenhang mit bayesschen Methoden sind drei Punkte zu beachten:

- Bayessche Methoden — wie klassische Netze auch — setzen voraus, dass es mindestens einen Gewichtsvektor w mit nicht verschwindender a priori Wahrscheinlichkeit gibt, sodass $g(x, w) \approx f(x)$ für alle Eingangsvektoren x aus dem betrachteten Raum gilt. Die Abweichung der beiden Funktionen sollte dabei deutlich kleiner als die Messfehler sein, wenn man sinnvolle Prognosefehler durch die bayesschen Methoden berechnen lassen möchte.
  Basieren nun die white boxes und/oder die Struktur des Modells nicht auf exaktem Wissen, sondern nur auf Näherungen oder Annahmen, so werden die bayesschen Methoden zu kleine Prognosefehler berechnen. Dies liegt daran, dass die Menge der möglichen wahren Funktionen, also die Menge der Funktionen, die dem Netz zur Verfügung steht, zu klein ist. Daraus folgt, dass natürlich auch die Verteilung von Prognosewerten t zu klein ist, bzw. die Variable t zu scharf bestimmt ist.
  In der Praxis muss in Fällen ungenauen analytischen Wissens natürlich ein Kompromiss zwischen einer zu einschränkenden Netzfunktion auf der einen Seite und dem Aufweichen/Ignorieren des analytischen Wissens auf der anderen Seite gefunden werden.

- Bei Hybridmodellen kann es eine sehr große Anzahl verschiedener Modelle geben, wenn man jeder black box eine individuelle Anzahl von Gewichten und/oder individuelle a priori Gewichtsverteilungen zuordnen möchte. Dieses Problem stellt sich zwar prinzipiell auch bei klassischen Netzen, allerdings begnügt man sich hier oft mit einer festen Wahl. Dem gegenüber ist eine der Stärken bayesscher Methoden gerade die Bewertung von verschiedenen Modellen.
  Wie bereits in Abschnitt 2.6.1 erwähnt, kann dieses Problem auch die bayesschen Methoden überfordern, wenn die Anzahl der Hyperparameter allzu groß ist.

- Der höhere Grad an Kompliziertheit der Netzfunktion kann zu Problemen beim Training führen. Dazu betrachte man die Funktion $S(w)$ (siehe Gleichung 2.8): mit der Kompliziertheit wächst die Gefahr, ein lokales, aber nicht globales Minimum w_{MP} zu finden. Außerdem kann die quadratische Approximation der Funktion $S(w)$ nach Gleichung 2.9 so schlecht werden, dass die nachfolgenden Berechnungen der Evidenz und der Prognosefehler inakzeptabel ungenau werden. Zu Fragen der Güte der Approximation äußern sich [MülIns] und [Thodberg].

2.6.4 Besondere Fehlerfunktionen

In Abschnitt 2.2 wurde für die Verteilung der beobachteten Werte (Trainingswerte) eine Normalverteilung um den wahren Wert angenommen (2.3). Für spezielle Daten können aber andere Verteilungen sinnvoll sein.

Als ein Beispiel sollen hier mögliche Ablese- bzw. Eingabefehler, verursacht durch Menschen, betrachtet werden. Wir nehmen an, dass beobachtete Werte von einem Messgerät abgelesen und dann über eine Tastatur in eine Datenbank eingegeben werden. Dabei unterläuft dem Datenerfasser mit der Wahrscheinlichkeit $P_{\text{Fehler}} > 0$ ein Ablese- oder Eingabefehler. In jedem Fall werden die eingegebenen Werte aber von der Datenbank geprüft und zurückgewiesen, falls sie sich nicht im gültigen Wertebereich befinden, der hier beispielhaft mit $[0, 100]$ angenommen wird. Im Falle eines Fehlers wird vereinfachend angenommen, dass jeder Wert im Wertebereich mit gleicher Wahrscheinlichkeit eingegeben wurde. Die Wahrscheinlichkeitsdichte nach Gleichung 2.4 wird nun durch die Dichte

$$p(t_n|w) = \left(1 - P_{\text{Fehler}}\right) \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t_n - g(x_n, w))^2\right) + P_{\text{Fehler}} \frac{1}{100} \tag{2.34}$$

ersetzt.

Die Trainingsdatenwahrscheinlichkeit nimmt nun die Form

$$p(D|w) = \prod_{n=1}^{N} \left(\left(1 - P_{\text{Fehler}}\right) \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t_n - g(x_n, w))^2\right) + P_{\text{Fehler}} \frac{1}{100} \right) \tag{2.35}$$

an, die daraus folgende Funktion $S(w)$ ist nun sehr kompliziert, was zu einem deutlich erhöhten Aufwand für die Optimierung führt. Es ist also theoretisch durchaus möglich, spezielle Messdatenverteilungen zu wählen und bayessche Methoden darauf anzuwenden, allerdings können sie zu nicht mehr handhabbaren Problemen bei der Implementierung führen.

An dieser Stelle soll aber darauf hingewiesen werden, dass auch bei klassischen Netzen unterschiedliche Fehlerfunktionen diskutiert werden. Diese korrespondieren in der Regel mit bestimmten Wahrscheinlichkeitsdichten für die Trainingsdaten ([Williams]).

2.7 Äquivalenz von Netzen

Ein Netz mit bayesschen Methoden ist eindeutig durch die Netzfunktion und die a priori Verteilung der Gewichte gegeben. Dies gilt jedoch nicht umgekehrt: es gibt Netze mit unterschiedlichen Netzfunktionen und unterschiedlichen a priori Gewichtsverteilungen, die aber identische Prognosen berechnen. Betrachten wir beispielhaft das Netz, das durch

$$\begin{aligned}
g(x, w) &= (w_3 + w_4) \tanh(w_1 x_1 + w_2 x_2) && (2.36)\\
p(w) &= p(w_1)p(w_2)p(w_3)p(w_4) && (2.37)\\
& w_1 \in [0, 2] \text{ gleichverteilt} && (2.38)\\
& w_2 \in [1, 2] \text{ gleichverteilt} && (2.39)\\
& w_3, w_4 \propto \mathcal{N}(0, 1) && (2.40)
\end{aligned}$$

gegeben ist. Man kann nun die Gewichte wie folgt transformieren: w_1 wird halbiert, w_2 wird um 1 verkleinert und w_3 und w_4 werden als Summe zusammengefasst. Man erhält dann das Netz

$$\begin{aligned}
g(x, w) &= w_3 \tanh(2w_1 x_1 + (w_2 + 1)x_2) && (2.41)\\
p(w) &= p(w_1)p(w_2)p(w_3) && (2.42)\\
& w_1, w_2 \in [0, 1] \text{ gleichverteilt} && (2.43)\\
& w_3 \propto \mathcal{N}(0, 2), && (2.44)
\end{aligned}$$

das die gleichen Prognosen berechnet wie das ursprüngliche.

Im Folgenden werden derartige Transformationen in den Gewichten allgemein untersucht. Dazu wird die Prognoseverteilung eines Netzes betrachtet, sie wird durch Gleichung 2.11 beschrieben. Setzt man dort die bayessche Gleichung, Gleichung 2.4 für die Verteilung einer Messung an der Stelle der Prognose

und Gleichung 2.5 ein, erhält man

$$\begin{aligned} p(t|x,D) &= \int_W p(t|x,w)\frac{p(w)}{p(D)}p(D|w)\,dw \\ &= \int_W \sqrt{\frac{\beta}{2\pi}}\exp\left(-\frac{\beta}{2}(t-g(x,w))^2\right)\frac{p(w)}{p(D)}\left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left(-\frac{\beta}{2}\sum_{n=1}^{N}(t_n-g(x_n,w))^2\right)dw \\ &= \int_W \frac{p(w)}{p(D)}\left(\frac{\beta}{2\pi}\right)^{\frac{N+1}{2}}\exp\left(-\frac{\beta}{2}\left((t-g(x,w))^2+\sum_{n=1}^{N}(t_n-g(x_n,w))^2\right)\right)dw, \end{aligned} \tag{2.45}$$

wobei $W \subseteq \mathbb{R}^m$ die Menge der Gewichte mit nicht verschwindender a priori Wahrscheinlichkeitsdichte ist. Durch diese Gleichung sind die Prognosen eines Netzes eindeutig bestimmt und in Abhängigkeit der Trainingsdaten D, der a priori Verteilung der Gewichte $p(w)$ und der Netzfunktion g beschrieben. Der Wert von β wird hier als gegeben angenommen, die Konstante $p(D)$ dient nur der Normierung.

Wir betrachten zwei Netze als äquivalent, wenn sie für beliebige Trainingsdaten und beliebige Prognosestellen identische Prognoseverteilungen berechnen. Wir zeigen nun, dass bestimmte Transformationen der Gewichte zu äquivalenten Netzen führen. Sei $h : \tilde{W} \to W$ eine $\mathcal{C}^1$-invertierbare[6] Abbildung mit $\tilde{W}, W \subseteq \mathbb{R}^M$, die ein Gewicht $\tilde{w}$ in ein Gewicht w transformiert. Dann gilt nach der Transformationsformel für integrierbare Funktionen (siehe dazu etwa [Forster]):

$$\begin{aligned} p(t|x,D) &= \int_{\tilde{W}} \frac{p(h(\tilde{w}))}{p(D)}\left(\frac{\beta}{2\pi}\right)^{\frac{N+1}{2}}\exp\left(-\frac{\beta}{2}\left((t-g(x,h(\tilde{w})))^2+\sum_{n=1}^{N}(t_n-g(x_n,h(\tilde{w})))^2\right)\right) \\ &\qquad \cdot\left|\det\frac{\partial h}{\partial\tilde{w}}(\tilde{w})\right| d\tilde{w}, \end{aligned} \tag{2.46}$$

wobei $\frac{\partial h}{\partial\tilde{w}}(\tilde{w})$ die Jacobi-Matrix der Funktion h an der Stelle $\tilde{w}$ ist. Man kann nun eine neue Netzfunktion $\tilde{g}$ und eine neue a priori Verteilung der Gewichte $\tilde{p}(\tilde{w})$ angeben,

$$\tilde{g}(x,\tilde{w}) := g(x,h(\tilde{w})) \tag{2.47}$$

$$\tilde{p}(\tilde{w}) := p(h(\tilde{w}))\cdot\left|\det\frac{\partial h}{\partial\tilde{w}}(\tilde{w})\right|, \tag{2.48}$$

sodass das neue Netz identische Prognosen berechnet.

Die praktischen Anwendungsmöglichkeiten einer derartigen Gewichtstransformation sind sehr begrenzt. Man ist in der Regel bemüht, einfache Netzfunktionen und einfache a priori Verteilungen zu verwenden, um die Algorithmen zum Training und zur numerischen Berechnung von Prognosen einfach und effizient zu gestalten. Eine Transformation wird in der Praxis also eher zu komplizierteren Berechnungen führen.

Die Menge der Funktionen, die das Netz darstellen kann, ist durch die Menge $\{g(.,w) \mid p(w) > 0\}$ gegeben. Diese Menge bleibt auch nach der Gewichtstransformation durch h gleich. Zu einem vorgegebenen Netz kann man also in der Regel kein äquivalentes Netz mit vorgegebener Netzfunktion, also durch Wahl der a priori Gewichtsverteilung, konstruieren. Umgekehrt kann man aber zu einem vorgegebenen Netz ein äquivalentes Netz mit vorgegebener a priori Gewichtsverteilung finden, wenn die Funktionalgleichung 2.48 in h lösbar ist. Dies dürfte nach Meinung des Autors in der Regel der Fall sein, wenn beide a priori Gewichtsverteilungen stetig sind. Die analytische Darstellung und damit die praktische Verwendbarkeit ist allerdings fraglich. Man sieht aber, dass die beiden Teile Netzfunktion und a priori Gewichtsverteilung unterschiedlich mächtigen Einfluss auf das Netz haben.

[6] Die Funktion h ist eine Bijektion von $\tilde{W}$ nach W und sowohl h als auch die Umkehrfunktion h^{-1} sind einmal stetig differenzierbar.

2.8 Zusammenfassung der Eigenschaften bayesscher Methoden

Wir haben gesehen, dass bayessche Methoden einerseits einen fundamental anderen Ansatz für Generalisierungsprobleme mit neuronalen Netzen darstellen als klassische Methoden, dass aber andererseits die resultierenden Verfahren sehr eng mit den Verfahren bei klassischen Methoden korrespondieren.

Hier einige korrespondieren Eigenschaften in der Übersicht:

Bayessche Methoden	Klassische Methoden
Modell	
Messfehlerrauschen	Datenterm der Fehlerfunktion
Normalverteilung des Messfehlers	quadratischer Datenterm der Fehlerfunktion
a priori Verteilung der Gewichte	Gewichtsregularisierung
a priori Normalverteilung der Gewichte	quadratische Gewichtsregularisierung
Verfahren	
Bestimmung des a posteriori wahrscheinlichsten Gewichtsvektors	Minimierung der Fehlerfunktion
Bestimmung des Erwartungswerts der Prognose	Evaluierung der Netzfunktion mit dem optimalen Gewichtsvektor
Approximation eines Komitees verschiedener Netzstrukturen durch das evidenteste Mitglied	Optimierung der Netzstruktur

Darüber hinaus bieten bayessche Methoden aber eine ganze Reihe von Vorteilen gegenüber klassischen Methoden:

- Bayessche Methoden stellen einen wesentlich besseren Modellierungsansatz dar, der die meisten praktischen Generalisierungsprobleme besser beschreibt. Dies ermöglicht eine bessere theoretische Durchleuchtung des gesamten Generalisierungsverfahrens.
- Es können neben den Prognosewerten auch Prognosefehler berechnet werden.
- Es wird keine Test- oder Validierungsdatenmenge benötigt. Dies führt in der Praxis dazu, dass mehr Daten, also mehr Wissen, in das trainierte Netz einfließen.
- Auch alle Hyperparameter können allein aufgrund der Trainingsdaten bestimmt werden. Es können zwar nicht beliebig viele, aber deutlich mehr Hyperparameter verwendet werden als bei klassischen Methoden.
- Es können (unter der Einschränkung normierbarer a priori Verteilungen) beliebige Netzmodelle miteinander verglichen werden. Dies beinhaltet vor allem verschiedene Anzahlen von Gewichten und verschiedene Netzstrukturen.
- Das konkrete Modell kann durch die größere Anzahl von Komponenten (Datenfehler, Netzfunktion, a priori Verteilung, Kombination von verschiedenen Netzstrukturen) besser an die konkreten Bedürfnisse eines Generalisierungsproblems angepasst werden.

Da die bayesschen Methoden als Verallgemeinerung bestimmter klassischer Verfahren angesehen werden können, werfen sie keine zusätzlichen Probleme auf; allerdings werden durch sie einige Probleme sichtbar, die aber implizit auch bei klassischen Verfahren vorhanden sind. Die Erweiterungen gegenüber klassischen Methoden können — wenn sie genutzt werden sollen — aber durchaus zu neuen Problemen führen: etwa die Qualität der Approximation der Hesse-Matrix A, die Einfluss auf die Berechnung der Evidenzen und der Prognosefehler hat.

Kapitel 3

Generalisierte lineare Netze mit expliziten Trainingsfehlern

In diesem Kapitel werden die der gesamten Arbeit zugrunde liegenden künstlichen neuronalen Netze mit bayesschen Methoden detailliert beschrieben. Ihre Eigenschaften werden genutzt, um Prognosen diverser Größen zu berechnen.

3.1 Definition der Netze

Sei $f : \mathbb{R}^L \to \mathbb{R}$ die unbekannte wahre Funktion. Sie beschreibt das zugrunde liegende Phänomen — im vorliegenden Fall die Korrosion — und ordnet jedem Eingangsvektor x einen eindeutigen Funktionswert $f(x)$ zu.

Ein Trainingsdatensatz entspricht einer Beobachtung und ist gegeben durch ein Tripel (x_n, t_n, s_n). Dabei ist $x_n \in \mathbb{R}^L$ der Eingangsvektor (im Folgenden auch Messstelle genannt), $t_n \in \mathbb{R}$ der beobachtete Wert (im Folgenden Trainingswert genannt) und $s_n \in \mathbb{R}^+$ die bekannte Standardabweichung des Messrauschens (im Folgenden Messfehler genannt). Es ist also

$$t_n \propto \mathcal{N}\left(f(x_n), s_n^2\right). \tag{3.1}$$

Insgesamt wurden N stochastisch unabhängige Messungen durchgeführt, somit ist die Menge der Trainingsdaten

$$D \quad = \quad \{(x_1, t_1, s_1), \ldots, (x_N, t_N, s_N)\}\,. \tag{3.2}$$

Da die wahre Funktion unbekannt ist, betrachtet man eine Menge von Funktionen $g(x, w)$ für Gewichtsvektoren $w \in \mathbb{R}^M$, die mögliche Kandidaten der Funktion $f(x)$ darstellen. Die Gewichtsvektoren unterliegen einer a priori Verteilung, die durch ihre Dichte $p(w)$ gegeben ist, und die dann einer a priori Verteilung der Funktionskandidaten entspricht.

Die Netzfunktion $g(x, w)$ ist eine sogenannte generalisierte lineare Funktion, die durch

$$g(x, w) \quad = \quad \sum_{m=1}^{M} g_m(x) w_m \tag{3.3}$$

oder in Vektorschreibweise $g(x, w) = g(x)^T w$ bestimmt ist. Die Funktionen $g_m : \mathbb{R}^L \to \mathbb{R}$ für $m = 1, \ldots, M$ werden Basisfunktionen genannt und hier zunächst nicht näher spezifiziert.

Unter der Annahme, dass ein gegebener Gewichtsvektor w der wahre ist, also $g(x, w) = f(x)$ für alle Eingangsvektoren x des betrachteten Raums gilt, ergibt sich die Trainingsdatenwahrscheinlichkeit zu

$$p(D|w) \quad = \quad \prod_{n=1}^{N} p(t_n|w)$$

$$
\begin{aligned}
&= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi s_n^2}} \exp\left(-\frac{(t_n - g(x_n, w))^2}{2s_n^2}\right) \\
&= \left(\prod_{n=1}^{N} \frac{1}{\sqrt{2\pi s_n^2}}\right) \cdot \exp\left(-\frac{1}{2}\sum_{n=1}^{N} \frac{(t_n - g(x_n)^T w)^2}{s_n^2}\right). \qquad (3.4)
\end{aligned}
$$

Dabei wurde ausgenutzt, dass die beobachteten Trainingswerte stochastisch unabhängig voneinander ermittelt wurden. Da die Messstellen $x_1, \ldots, x_N$ und die Messfehler $s_1, \ldots, s_N$ fest vorgegeben und keine Zufallsvariablen sind, werden sie bei der Notation von Wahrscheinlichkeiten und -dichten weggelassen.

Die Gewichte sollen a priori stochastisch unabhängig voneinander verteilt sein. Jedes einzelne Gewicht soll normalverteilt sein und den Erwartungswert 0 und die Standardabweichung $\sigma_w \in \mathbb{R}^+$ besitzen. Die Variable σ_w ist dabei der einzige Hyperparameter des Modells. Es ist

$$
\begin{aligned}
p(w|\sigma_w) &= \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{w_m^2}{2\sigma_w^2}\right) \\
&= \left(\frac{1}{2\pi\sigma_w^2}\right)^{M/2} \cdot \exp\left(-\frac{1}{2\sigma_w^2}\sum_{m=1}^{M} w_m^2\right). \qquad (3.5)
\end{aligned}
$$

Die a posteriori Verteilung der Gewichte ergibt sich nun für ein vorgegebenes σ_w durch die bayessche Gleichung:

$$
\begin{aligned}
p(w|D, \sigma_w) &= \frac{p(w|\sigma_w)p(D|w, \sigma_w)}{p(D|\sigma_w)} \\
&= \frac{p(w|\sigma_w)p(D|w)}{p(D)} \\
&= \frac{1}{p(D)}\left(\frac{1}{2\pi\sigma_w^2}\right)^{M/2}\left(\prod_{n=1}^{N}\frac{1}{\sqrt{2\pi s_n^2}}\right) \\
&\qquad \cdot \exp\left(-\sum_{m=1}^{M}\frac{w_m^2}{2\sigma_w^2} - \sum_{n=1}^{N}\frac{(t_n - g(x_n)^T w)^2}{2s_n^2}\right). \qquad (3.6)
\end{aligned}
$$

Die aus Gleichung 2.8 bekannte Funktion S wird als

$$
\begin{aligned}
S(w) &:= \sum_{m=1}^{M}\frac{w_m^2}{2\sigma_w^2} + \sum_{n=1}^{N}\frac{(t_n - g(x_n)^T w)^2}{2s_n^2} \qquad (3.7) \\
&= -\ln p(w|D, \sigma_w) + const
\end{aligned}
$$

gewählt. Man beachte, dass hier der Zusammenhang mit der a posteriori Dichte der Gewichte exakt und nicht nur approximativ gilt: hierin liegt ein wesentlicher Vorteil generalisierter linearer Netzfunktionen. Die Funktion S ist quadratisch in den Gewichten und kann entsprechend umgeformt werden:

$$
\begin{aligned}
S(w) &= \frac{1}{2}\left(\frac{w^T w}{\sigma_w^2} + \sum_{n=1}^{N}\frac{(t_n - g(x_n)^T w)^2}{s_n^2}\right) \\
&= \frac{1}{2}\left(\frac{w^T I w}{\sigma_w^2} + \sum_{n=1}^{N}\frac{t_n^2 - 2t_n g(x_n)^T w + w^T g(x_n) g(x_n)^T w)}{s_n^2}\right) \\
&= \frac{1}{2}\left(w^T \underbrace{\left(\frac{1}{\sigma_w^2} I + \sum_{n=1}^{N}\frac{1}{s_n^2} g(x_n) g(x_n)^T\right)}_{A} w - 2\underbrace{\left(\sum_{n=1}^{N}\frac{t_n}{s_n^2} g(x_n)^T\right)}_{b^T} w + \underbrace{\sum_{n=1}^{N}\frac{t_n^2}{s_n^2}}_{c}\right). \qquad (3.8)
\end{aligned}
$$

Die Matrix A entspricht der aus Abschnitt 2.2, sie ist symmetrisch und positiv definit. Mit den Abkürzungen A, b und c folgt weiter

$$\begin{aligned} S(w) &= \frac{1}{2}\left(w^T A w - 2b^T w + c\right) \\ &= \frac{1}{2}\left(w^T A w - b^T A^{-1} A w - w^T A A^{-1} b + c\right) \\ &= \frac{1}{2}\left((w - A^{-1}b)^T A (w - A^{-1}b) - b^T A^{-1} b + c\right) \\ &= \frac{1}{2}(w - w_{\mathrm{MP}})^T A (w - w_{\mathrm{MP}}) + const, \end{aligned} \tag{3.9}$$

wobei $w_{\mathrm{MP}} := A^{-1}b$ festgelegt wurde.

Da S quadratische Form in w hat, muss $p(w|D, \sigma_{\mathrm{w}})$ die Dichte einer Normalverteilung sein. Der Vorfaktor dieser Dichte ergibt sich eindeutig, da die a posteriori Dichte der Gewichte normierbar sein muss. Sie lautet

$$\begin{aligned} p(w|D, \sigma_{\mathrm{w}}) &= const \cdot \exp(-S(w)) \\ &= \sqrt{\frac{\det A}{(2\pi)^M}} \cdot \exp\left(-\frac{1}{2}(w - w_{\mathrm{MP}})^T A (w - w_{\mathrm{MP}})\right) \end{aligned} \tag{3.10}$$

$$w|D, \sigma_{\mathrm{w}} \propto \mathcal{N}\left(w_{\mathrm{MP}}, A^{-1}\right). \tag{3.11}$$

Aus dieser Verteilung der Gewichte kann für eine Prognoseanfrage an der Stelle x direkt die Verteilung der Ausgangsvariablen $t = g(x)^T w$ bestimmt werden ([Müller]):

$$t|D, \sigma_{\mathrm{w}}, x \propto \mathcal{N}\left(g(x)^T w_{\mathrm{MP}}, g(x)^T A^{-1} g(x)\right). \tag{3.12}$$

Die zu berechnenden Ausgaben sind die Kenngrößen dieser Verteilung:

$$\begin{aligned} \mu(x) &:= E[t|D, \sigma_{\mathrm{w}}, x] \qquad \text{(Prognosewert)} \\ &= g(x)^T w_{\mathrm{MP}} \end{aligned} \tag{3.13}$$

$$\begin{aligned} \sigma^2(x) &:= VAR[t|D, \sigma_{\mathrm{w}}, x] \qquad \text{(Prognosevarianz)} \\ &= g(x)^T A^{-1} g(x) \end{aligned} \tag{3.14}$$

Die Prognosevarianz unterscheidet sich von der Verteilung nach Gleichung 2.14 durch das Fehlen des Summanden β^{-1}. Der zu bestimmende Hyperparameter β, der die Stärke des Rauschens aller Trainingsdaten modelliert, wurde jedoch im Modell hier durch die bekannten expliziten und individuellen Messfehler $s_1, \ldots, s_N$ ersetzt und ist somit nicht verfügbar. Die Prognose hier beschreibt die Lage des wahren Funktionswerts $f(x)$ während die Prognose in Abschnitt 2.3 die Lage des Messwerts einer Testmessung an der Stelle x beschreibt.

Die hier eingeführten Symbole der Trainingsdaten und Netzgrößen werden durchgängig in der gesamten weiteren Arbeit verwendet. Sie sind daher tabellarisch in Anhang A auf Seite 161 aufgeführt.

3.1.1 Algorithmische Umsetzung

Für eine gegebene a priori Standardabweichung σ_{w} der Gewichte besteht das Training aus folgenden Schritten:

1. Berechne A und b nach Gleichung 3.8 (Laufzeit[1] $O(NM^2)$).
2. Invertiere A, berechne $w_{\mathrm{MP}} = A^{-1}b$ (Laufzeit $O(M^3)$).
3. Speichere w_{MP} und A^{-1} als Ergebnis des Trainings (Speicherplatz $O(M^2)$).

[1] Die Anzahl der Eingänge wird als konstant angenommen, dadurch kann eine Basisfunktion in konstanter Zeit berechnet werden. Die angegebenen Laufzeiten sind mit bekannten numerischen Algorithmen ([PreTeuVet]) realisierbar.

Man beachte, dass alle Berechnungen im Rahmen der Numerik exakt durchgeführt werden können, es sind keine iterativen Berechnungen nötig.

Bei der Prognose werden folgende Schritte durchgeführt:

1. Berechne die Basisfunktionen $g(x)$ (Laufzeit $O(M)$).
2. Berechne $\mu(x) = g(x)^T w_{\mathrm{MP}}$ (Laufzeit $O(M)$).
3. Berechne $\sigma^2(x) = g(x)^T A^{-1} g(x)$ (Laufzeit $O(M^2)$).

Ist man an Prognosefehlern nicht interessiert, so kann man sich die Speicherung der Matrix A^{-1} sparen, die benötigte Speicherplatzgröße reduziert sich dann auf $O(M)$. Außerdem reduziert sich die Laufzeit der Prognose auf ebenfalls $O(M)$.

3.2 Bestimmung des Hyperparameters σ_{w}

Um das Modell aus Abschnitt 3.1 zu vervollständigen, ist es nötig, die a posteriori Verteilung des einzigen Hyperparameters σ_{w} zu bestimmen. Für die Prognose ist allein die a posteriori Verteilung der Gewichte entscheidend, für die

$$p(w|D) \quad = \quad \int p(w|D, \sigma_{\mathrm{w}}) p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}} \tag{3.15}$$

gilt. Leider hängen die Ausdrücke $p(w|D, \sigma_{\mathrm{w}})$ und $p(\sigma_{\mathrm{w}}|D)$ auf komplizierte Weise von σ_{w} ab, sodass eine analytische Lösung des Integrals nicht gefunden werden konnte[2].

Eine Methode das Integral in 3.15 zu berechnen wäre eine Beschreibung der Verteilung $\sigma_{\mathrm{w}}|D$ durch eine repräsentative Menge von Stichproben $\sigma_{\mathrm{w}}^{(1)}, \ldots, \sigma_{\mathrm{w}}^{(J)}$ aus dieser Verteilung:

$$\begin{aligned} p(w|D) &= \frac{1}{J} \sum_{j=1}^{J} p(w|D, \sigma_{\mathrm{w}}^{(j)}) \\ &= \frac{1}{J} \sum_{j=1}^{J} \sqrt{\frac{\det A(\sigma_{\mathrm{w}}^{(j)})}{(2\pi)^M}} \cdot \exp\left(-\frac{1}{2} \left(w - w_{\mathrm{MP}}(\sigma_{\mathrm{w}}^{(j)}) \right)^T A(\sigma_{\mathrm{w}}^{(j)}) \left(w - w_{\mathrm{MP}}(\sigma_{\mathrm{w}}^{(j)}) \right) \right), \end{aligned} \tag{3.16}$$

es entsteht eine Mixturverteilung. Diese Vorgehensweise entspricht einem Komitee, jedes Komiteemitglied besitzt einen scharfen Wert für σ_{w} und das gesamte Komitee deckt die Verteilung von σ_{w} ab.

Leider würde eine direkte Implementierung dieses Verfahrens bei einem Komitee mit J Mitgliedern die Prognoselaufzeit und den Speicheraufwand für das Komitee um den Faktor J gegenüber den entsprechenden Größen aus Abschnitt 3.1.1 vergrößern. Daher wurde für die Implementierung eine Komiteegröße von Eins gewählt. Die einzige Stichprobe $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ muss daher sorgfältig gewählt werden, um die gesamte Verteilung möglichst gut zu beschreiben. Ein mögliches Problem bei dieser Wahl zeigt der folgende Abschnitt in Form eines sehr einfachen Beispiels auf.

3.2.1 Ein Rechenbeispiel

Das folgende Beispiel ist so einfach gehalten, dass alle Größen und Verteilungen analytisch berechnet werden können. Die gesuchte wahre Funktion f ist eine Konstante, d.h. sie hängt nicht von einer Messstelle ab ($L = 0$). Entsprechend einfach ist die Netzfunktion $g(w) = w$, wobei der Gewichtsraum eindimensional ist. Die Trainingsdaten bestehen aus nur einer Messung mit Messwert t und Messfehler s.

[2] Es ist durchaus möglich, dass eine analytische Lösung des Integrals gefunden werden kann. Die Suche nach ihr, die zu einem nicht-iterativen Algorithmus führen könnte, ist unter Verwendung der Eigenwertmethode nach Abschnitt 3.2.4 im Hinblick auf die Rechenzeit des Trainings allerdings kaum von praktischem Nutzen.

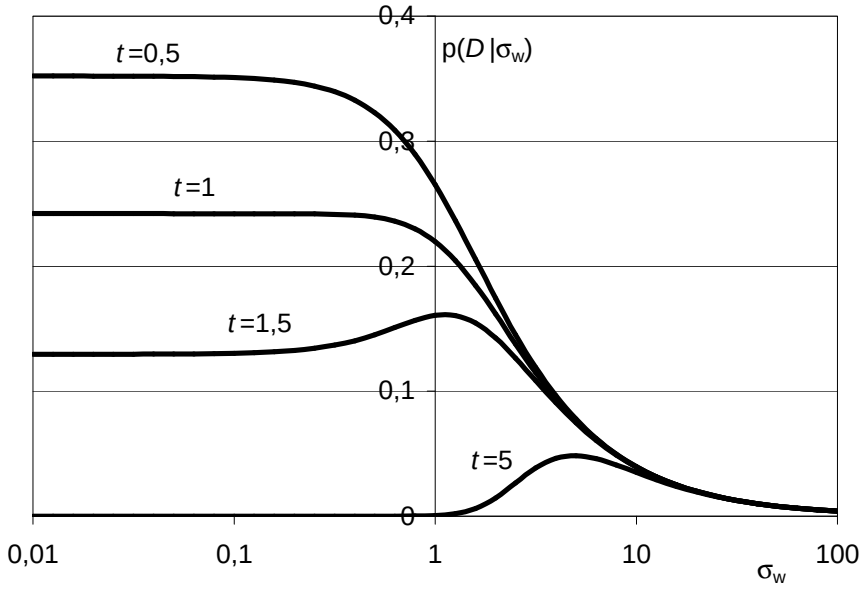

Abbildung 3.1: Verlauf der Evidenz nach Gleichung 3.19 für $s = 1$.

Daraus folgen die Verteilungen

$$p(D|w) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{(t-w)^2}{2s^2}\right) \qquad \text{und} \tag{3.17}$$

$$p(w) = \frac{1}{\sqrt{2\pi\sigma_{\mathrm{w}}^2}} \exp\left(-\frac{w^2}{2\sigma_{\mathrm{w}}^2}\right), \tag{3.18}$$

woraus sich die Evidenz bezüglich eines festen Wertes von σ_{w} bestimmen lässt:

$$\begin{aligned}
p(D|\sigma_{\mathrm{w}}) &= \int p(D|w)p(w)\,dw \\
&= \int \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\left(\frac{(t-w)^2}{s^2} + \frac{w^2}{\sigma_{\mathrm{w}}^2}\right)\right) dw \\
&= \int \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\left(s^{-2}t^2 - 2s^{-2}tw + s^{-2}w^2 + \sigma_{\mathrm{w}}^{-2}w^2\right)\right) dw \\
&= \int \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\left(\sigma_{\mathrm{w}}^{-2} + s^{-2}\right)\left(w^2 - 2\frac{s^{-2}}{\sigma_{\mathrm{w}}^{-2} + s^{-2}}tw + \frac{s^{-2}}{\sigma_{\mathrm{w}}^{-2} + s^{-2}}t^2\right)\right) dw \\
&= \int \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\left(\sigma_{\mathrm{w}}^{-2} + s^{-2}\right)\left(\left(w - \frac{s^{-2}}{\sigma_{\mathrm{w}}^{-2} + s^{-2}}t\right)^2\right.\right. \\
&\qquad \left.\left. - \frac{s^{-4}}{(\sigma_{\mathrm{w}}^{-2} + s^{-2})^2}t^2 + \frac{s^{-2}}{\sigma_{\mathrm{w}}^{-2} + s^{-2}}t^2\right)\right) dw \\
&= \int \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\left(\sigma_{\mathrm{w}}^{-2} + s^{-2}\right)\left(w^2 + \frac{s^{-2}\sigma_{\mathrm{w}}^{-2}}{(\sigma_{\mathrm{w}}^{-2} + s^{-2})^2}t^2\right)\right) dw \\
&= \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_{\mathrm{w}}^2 + s^2}\right) \int \exp\left(-\frac{1}{2}\left(\sigma_{\mathrm{w}}^{-2} + s^{-2}\right)w^2\right) dw \\
&= \frac{1}{2\pi s\sigma_{\mathrm{w}}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_{\mathrm{w}}^2 + s^2}\right) \sqrt{\frac{2\pi}{\sigma_{\mathrm{w}}^{-2} + s^{-2}}}
\end{aligned}$$

$$= \frac{1}{\sqrt{2\pi(\sigma_w^2 + s^2)}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_w^2 + s^2}\right). \tag{3.19}$$

Abbildung 3.1 zeigt die Evidenz des Modells in Abhängigkeit der Standardabweichung σ_w der a priori Verteilung der Gewichte für verschiedene Messwerte t.

Den Zusammenhang zwischen dieser Evidenz und der in Gleichung 3.15 geforderten a posteriori Verteilung von σ_w erhält man über die bayessche Gleichung

$$p(\sigma_w|D) = \frac{p(D|\sigma_w)p(\sigma_w)}{p(D)}. \tag{3.20}$$

Es stellt sich hier die Frage nach der Wahl der a priori Verteilung von σ_w. Da σ_w ein Skalierungsparameter ist, liegt die Wahl $p(\sigma_w) = 1/\sigma_w$ nahe ([Berger]). Diese Wahl führt aber zu einer problematischen a posteriori Verteilung, wie durch die folgenden Betrachtungen gezeigt werden kann. Betrachten wir den Grenzwert

$$\lim_{\sigma_w \to 0} p(D|\sigma_w) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{t^2}{2s^2}\right) > 0. \tag{3.21}$$

Da $p(D|\sigma_w)$ stetig in σ_w ist, gibt es Zahlen $\sigma_w^0 > 0$ und $c > 0$, sodass $p(D|\sigma_w) \geq c$ für alle $\sigma_w \in [0, \sigma_w^0]$ gilt. Aus dieser unteren Schranke für die a priori Verteilung folgt aber für die a posteriori Dichte

$$\begin{aligned}
\int p(\sigma_w|D)\, d\sigma_w &= \int \frac{1}{p(D)} \frac{p(D|\sigma_w)}{\sigma_w}\, d\sigma_w \\
&\geq \frac{1}{p(D)} \int_0^{\sigma_w^0} \frac{p(D|\sigma_w)}{\sigma_w}\, d\sigma_w \\
&\geq \frac{1}{p(D)} \int_0^{\sigma_w^0} \frac{c}{\sigma_w}\, d\sigma_w.
\end{aligned} \tag{3.22}$$

Bekanntermaßen ist dieses Integral nicht beschränkt und es folgt, dass die a posteriori Dichte nicht normierbar ist.

Anschaulich bedeutet dieses Resultat, dass die Wahrscheinlichkeit, dass eine Stichprobe von σ_w außerhalb des Intervalls $[0, \sigma_w^0]$ liegt, immer Null beträgt, denn dort ist das entsprechende Integral beschränkt. Daraus folgt, dass eine repräsentative Stichprobe von σ_w nur aus infinitesimal kleinen Werten bestehen kann, da diese Aussage für jede beliebig kleine Zahl σ_w^0 gilt. Das Modell wählt also das einzige Gewicht $w = 0$: es nimmt dabei zwar einen großen, aber beschränkten Fehler zwischen trainiertem und gemessenem Wert in Kauf, gewinnt aber eine beliebig große Wahrscheinlichkeitsdichte bei der Wahl von σ_w.

Dieses erste Problem kann gelöst werden, indem die a priori Verteilung von σ_w als gleichverteilt für alle $\sigma_w > 0$ gewählt wird. Es ist ab hier also in Einklang mit der bayesschen Gleichung $p(\sigma_w|D) = const \cdot p(D|\sigma_w)$, wobei die Konstante *const* positiv ist.

Die Antwort auf die Frage, wie ein repräsentativer Wert σ_w^{opt} für die Zufallsvariable σ_w bestimmt werden kann, könnte die Bestimmung des wahrscheinlichsten Werts sein. So wird etwa in [Bishop] mit den Hyperparametern α und β verfahren. Den Wert σ_w^{opt} des wahrscheinlichsten σ_w erhält man leicht analytisch durch Differenzieren

$$\begin{aligned}
\frac{\partial}{\partial \sigma_w} p(\sigma_w|D) &= \frac{\partial}{\partial \sigma_w}\, const \cdot p(D|\sigma_w) \\
&= const \cdot \left(-\frac{2\sigma_w}{\sqrt{2\pi} \cdot 2(\sigma_w^2 + s^2)^{3/2}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_w^2 + s^2}\right) \right. \\
&\qquad \left. + \frac{1}{\sqrt{2\pi(\sigma_w^2 + s^2)}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_w^2 + s^2}\right) \cdot \frac{2t^2\sigma_w}{2(\sigma_w^2 + s^2)^2} \right) \\
&= const \cdot \frac{\sigma_w}{\sqrt{2\pi}(\sigma_w^2 + s^2)^{3/2}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_w^2 + s^2}\right) \left(-1 + \frac{t^2}{(\sigma_w^2 + s^2)}\right)
\end{aligned} \tag{3.23}$$

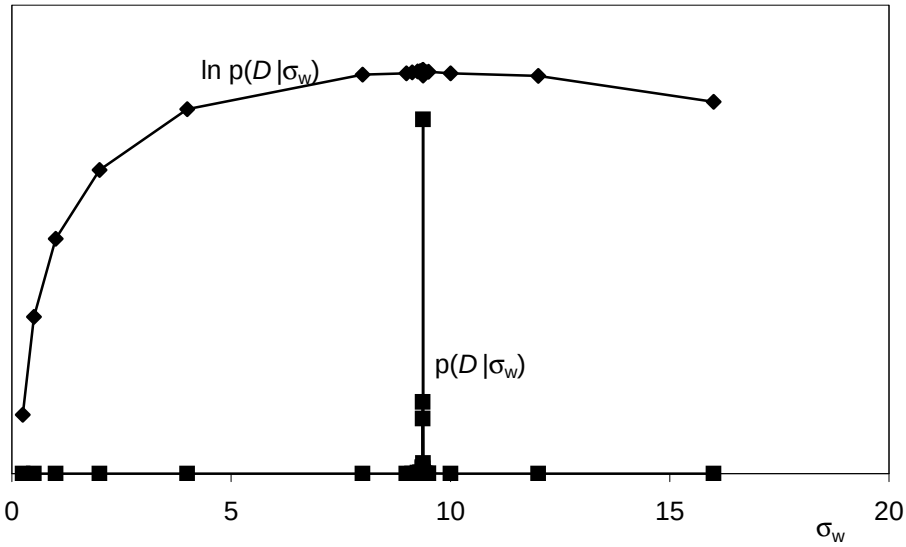

Abbildung 3.2: Evidenz eines realen Netzes zur Abtragungsgeschwindigkeit. Dargestellt sind die Evidenz und ihr Logarithmus auf verschiedenen Skalen in Abhängigkeit der Gewichtsregularisierung.

und Suche der Nullstellen. Offensichtlich sind die ersten drei Faktoren immer positiv und der vierte Faktor hat genau bei $\sigma_{\mathrm{w}}^{\mathrm{opt}} = \sqrt{t^2 - s^2}$ seine einzige Nullstelle.

Man sieht jedoch, dass $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ genau dann existiert und positiv ist, wenn $t > s$ ist. Sowohl t als auch s sind aber durch die Trainingsdaten vorgegeben. Im Fall $t \leq s$ ist also $p(\sigma_{\mathrm{w}}|D)$ für $\sigma_{\mathrm{w}} \geq 0$ monoton fallend und selbst das globale Maximum $\sigma_{\mathrm{w}}^{\mathrm{opt}} = 0$ ist nicht repräsentativ für die Verteilung von $\sigma_{\mathrm{w}}|D$. Diese Erkenntnis stellt ein zweites Problem dar: es ist offensichtlich nicht (immer) möglich $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ als das wahrscheinlichste σ_{w} zu definieren.

3.2.2 Bestimmung von σ_{w} über den Median

Aufgrund der in Abschnitt 3.2.1 für ein Beispiel beschriebenen Probleme wird nun $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ als Median der Verteilung von $\sigma_{\mathrm{w}}|D$ definiert. Alternativ dazu wäre beispielsweise auch die Wahl als Erwartungswert (bei Existenz) möglich.

Die Abbildungen 3.2, 3.3 und 3.4 stellen die Evidenz dreier Netze auf realen Daten dar. Es handelt sich dabei immer um die gleichen Trainingsdaten, allerdings wurden an manchen Messstellen nicht alle Ausgangswerte gemessen. Die Abtragungsgeschwindigkeit ist ein kontinuierlicher Parameter der Korrosion, der vergleichsweise oft und genau gemessen wurde. Daher besteht bei ihr kein signifikanter Unterschied zwischen dem Median und dem Maximum der Evidenz. Der Flächenabtrag und der Lochfraß basieren auf diskontinuierlichen Ausgangsparametern, die für das Netztraining vergleichsweise wenig Information (geringe Schwankung der Messwerte im Vergleich zu ihren Messfehlern) zur Verfügung stellen. Während beim Flächenabtrag ein erkennbarer, aber nicht wesentlicher Unterschied zwischen dem Median und dem Maximum besteht, liegt beim Lochfraß, dessen Trainingsdaten am wenigsten informativ waren, das Evidenzmaximum bei 0 und ist somit nicht verwendbar.

Die Verteilungsfunktion $F(\sigma_{\mathrm{w}}|D)$ der Zufallsvariablen $\sigma_{\mathrm{w}}|D$ kann aus der Wahrscheinlichkeitsdichtenfunktion $p(\sigma_{\mathrm{w}}|D)$ berechnet werden,

$$F(\sigma'_{\mathrm{w}}|D) \quad := \quad \int_0^{\sigma'_{\mathrm{w}}} p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}}, \tag{3.24}$$

und ist eine monoton steigende Funktion von $\mathbb{R}^+$ in das Intervall $]0, 1[$. Der Median $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ ist definiert

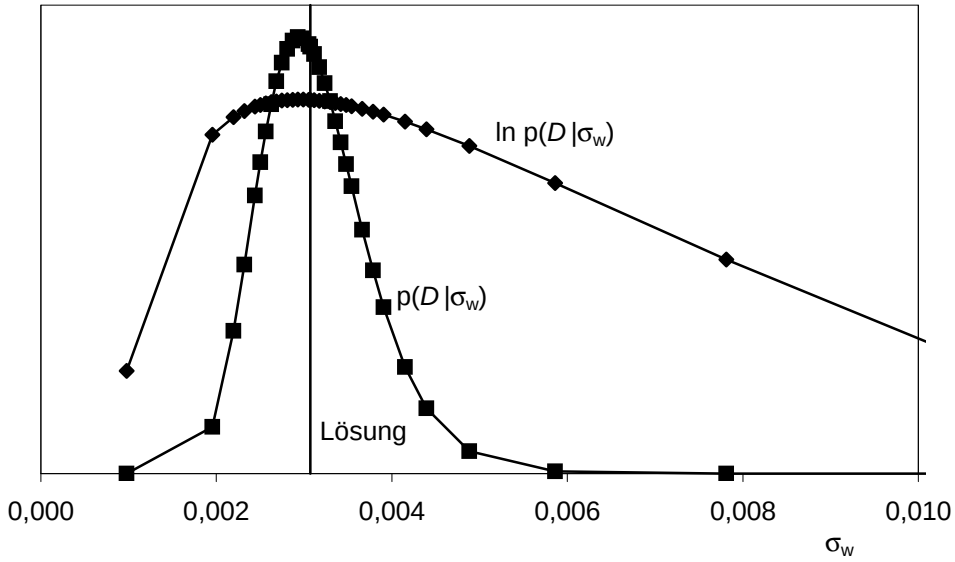

Abbildung 3.3: Evidenz eines realen Netzes zum Flächenabtrag. Die Kurvenmarker stellen die Stützstellen des Algorithmus dar und die „Lösung“ bezeichnet den gefundenen Median.

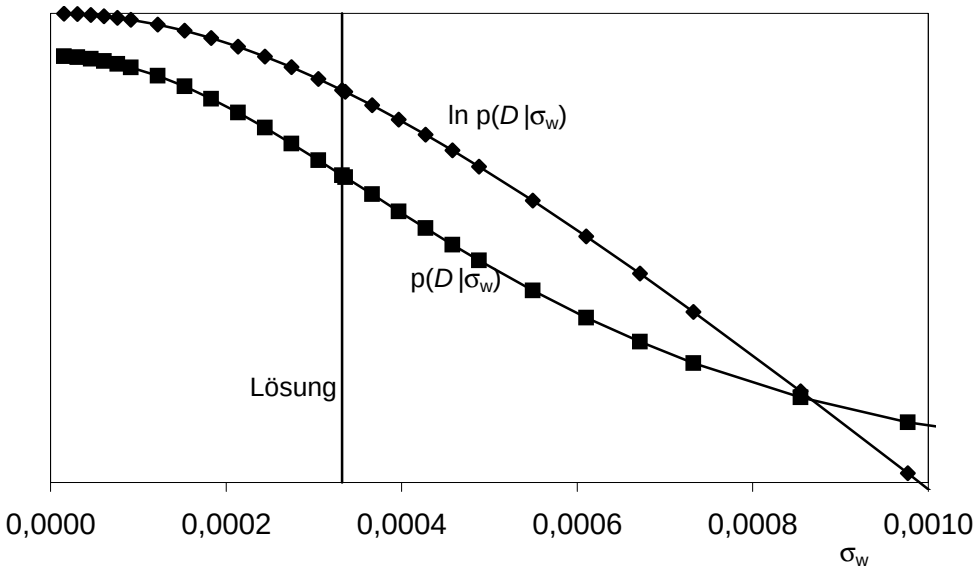

Abbildung 3.4: Evidenz eines realen Netzes zum Lochfraß.

durch $F(\sigma_{\mathrm{w}}^{\mathrm{opt}}|D) = 1/2$, er erfüllt daher die Bedingung

$$\int_0^{\sigma_{\mathrm{w}}^{\mathrm{opt}}} p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}} \quad = \quad \int_{\sigma_{\mathrm{w}}^{\mathrm{opt}}}^{\infty} p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}}. \tag{3.25}$$

Die Implementierung zur Berechnung von $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ evaluiert nun punktweise die Funktion $p(\sigma_{\mathrm{w}}|D)$. Aus den so verfügbaren Stützstellen berechnet ein numerisches Verfahren eine approximative Lösung der Gleichung 3.25.

Nach Gleichung 3.20 und der Festlegung der a priori Verteilung $p(\sigma_{\mathrm{w}}) = const$ basierend auf Abschnitt 3.2.1 kann die a posteriori Verteilung des Hyperparameters σ_{w} berechnet werden:

$$\begin{aligned}
p(\sigma_{\mathrm{w}}|D) &= const \cdot p(D|\sigma_{\mathrm{w}}) \\
&= const \cdot \int p(w|\sigma_{\mathrm{w}}) p(D|w)\, dw \\
&= const \cdot \int \left(\frac{1}{2\pi\sigma_{\mathrm{w}}^2}\right)^{M/2} \cdot \exp\left(-\frac{1}{2\sigma_{\mathrm{w}}^2}\sum_{m=1}^{M} w_m^2\right) \cdot \exp\left(-\frac{1}{2}\sum_{n=1}^{N}\frac{(t_n - g(x_n)^T w)^2}{s_n^2}\right) dw \\
&= const \cdot \left(\frac{1}{2\pi\sigma_{\mathrm{w}}^2}\right)^{M/2} \int \exp\left(-\frac{1}{2}\left(\frac{w^T w}{\sigma_{\mathrm{w}}^2} + \sum_{n=1}^{N}\frac{-2t_n g(x_n)^T w + w^T g(x_n) g(x_n)^T w}{s_n^2}\right)\right) dw \\
&= const \cdot \left(\frac{1}{2\pi\sigma_{\mathrm{w}}^2}\right)^{M/2} \int \exp\left(-\frac{1}{2} w^T \left(\frac{1}{\sigma_{\mathrm{w}}^2} I + \sum_{n=1}^{N}\frac{1}{s_n^2} g(x_n) g(x_n)^T\right) w \right. \\
&\qquad \left. + \left(\sum_{n=1}^{N}\frac{t_n}{s_n^2} g(x_n)^T\right) w\right) dw \\
&= const \cdot \left(\frac{1}{2\pi\sigma_{\mathrm{w}}^2}\right)^{M/2} \int \exp\left(-\frac{1}{2} w^T A w + b^T w\right) dw \qquad (3.26) \\
&= const \cdot \left(\frac{1}{2\pi\sigma_{\mathrm{w}}^2}\right)^{M/2} \cdot (2\pi)^{M/2} \frac{1}{\sqrt{\det A}} \exp\left(\frac{1}{2} b^T A^{-1} b\right) \\
&= const \cdot \sigma_{\mathrm{w}}^{-M} \frac{1}{\sqrt{\det A}} \exp\left(\frac{1}{2} b^T A^{-1} b\right) \qquad (3.27)
\end{aligned}$$

Die Berechnung des gaußschen Integrals in Zeile 3.26 findet sich etwa in [Bishop], appendix B.

Die benötigte a posteriori Dichte ist somit bis auf einen konstanten Faktor bestimmt und kann direkt berechnet werden. Da beim Training des Netzes ohnehin die Matrizen A und A^{-1} berechnet werden müssen, und sich die Determinante $\det A$ in Zeit $O(M^3)$ bestimmen lässt, bleibt durch die Berechnung dieser a posteriori Wahrscheinlichkeit die Gesamtlaufzeit zur Bestimmung einer Stützstelle bei σ_{w} bei $O((N+M)M^2)$.

Der konstante Faktor stellt kein prinzipielles Problem dar, denn Gleichung 3.25 gilt auch, wenn die Wahrscheinlichkeitsdichten beliebig skaliert werden: für jede Zahl $c > 0$ gilt

$$\int_0^{\sigma_{\mathrm{w}}^{\mathrm{opt}}} c \cdot p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}} \quad = \quad \int_{\sigma_{\mathrm{w}}^{\mathrm{opt}}}^{\infty} c \cdot p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}}, \tag{3.28}$$

wobei $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ nicht von c abhängt.

Vor der Beschreibung eines Algorithmus muss noch die allgemeine Existenz der beiden Integrale gezeigt werden; dies ist nötig, da die verwendete a priori Verteilung von σ_{w} nicht normierbar ist. Dazu sei der datenabhängige (σ_{w}-unabhängige) Teil der Matrix A mit

$$A_D \quad := \quad \sum_{n=1}^{N} 1/s_n^2 g(x_n) g(x_n)^T \tag{3.29}$$

abgekürzt, sodass $A = \sigma_{\mathrm{w}}^{-2} I + A_D$ gilt. Im Folgenden wird angenommen, dass A_D regulär ist. Die Abschätzung wird für ein beliebiges $\sigma_{\mathrm{w}}^{\mathrm{opt}} > 0$ vorgenommen, das nicht notwendigerweise Gleichung 3.25 erfüllen muss.

Der Integrand $p(\sigma_{\mathrm{w}}|D)$ in Gleichung 3.25 besitzt den Grenzwert

$$\begin{aligned}
\lim_{\sigma_{\mathrm{w}} \to 0} p(\sigma_{\mathrm{w}}|D) &= const \cdot \lim_{\sigma_{\mathrm{w}} \to 0} \sigma_{\mathrm{w}}^{-M} \frac{1}{\sqrt{\det(\sigma_{\mathrm{w}}^{-2} I + A_D)}} \exp\left(\frac{1}{2} b^T (\sigma_{\mathrm{w}}^{-2} I + A_D)^{-1} b\right) \\
&= const \cdot \lim_{\sigma_{\mathrm{w}} \to 0} \frac{\sigma_{\mathrm{w}}^{-M}}{\sqrt{\sigma_{\mathrm{w}}^{-2M} \det(I + \sigma_{\mathrm{w}}^2 A_D)}} \cdot \lim_{\sigma_{\mathrm{w}} \to 0} \exp\left(\frac{1}{2} b^T (\sigma_{\mathrm{w}}^{-2} (I + \sigma_{\mathrm{w}}^2 A_D))^{-1} b\right) \\
&= const \cdot \lim_{\sigma_{\mathrm{w}} \to 0} \frac{1}{\sqrt{\det(I + \sigma_{\mathrm{w}}^2 A_D)}} \cdot \lim_{\sigma_{\mathrm{w}} \to 0} \exp\left(\frac{\sigma_{\mathrm{w}}^2}{2} b^T (I + \sigma_{\mathrm{w}}^2 A_D)^{-1} b\right) \\
&= const \cdot 1 \cdot 1. \qquad (3.30)
\end{aligned}$$

Da der Integrand für $\sigma_{\mathrm{w}} \in]0, \sigma_{\mathrm{w}}^{\mathrm{opt}}]$ stetig und an beiden Grenzen endlich ist, ist er folglich beschränkt, und das Integral auf der linken Seite von Gleichung 3.25 existiert.

Die Existenz des Integrals auf der rechten Seite kann durch folgende Abschätzung gezeigt werden:

$$\begin{aligned}
\int_{\sigma_{\mathrm{w}}^{\mathrm{opt}}}^{\infty} c \cdot p(\sigma_{\mathrm{w}}|D)\, d\sigma_{\mathrm{w}} &= const \cdot \int_{\sigma_{\mathrm{w}}^{\mathrm{opt}}}^{\infty} \sigma_{\mathrm{w}}^{-M} \frac{1}{\sqrt{\det(\sigma_{\mathrm{w}}^{-2} I + A_D)}} \exp\left(\frac{1}{2} b^T (\sigma_{\mathrm{w}}^{-2} I + A_D)^{-1} b\right) d\sigma_{\mathrm{w}} \\
&\leq const \cdot \int_{\sigma_{\mathrm{w}}^{\mathrm{opt}}}^{\infty} \sigma_{\mathrm{w}}^{-M}\, d\sigma_{\mathrm{w}} \cdot \sup_{\sigma_{\mathrm{w}} \geq \sigma_{\mathrm{w}}^{\mathrm{opt}}} \frac{1}{\sqrt{\det(\sigma_{\mathrm{w}}^{-2} I + A_D)}} \\
&\quad \cdot \sup_{\sigma_{\mathrm{w}} \geq \sigma_{\mathrm{w}}^{\mathrm{opt}}} \exp\left(\frac{1}{2} b^T (\sigma_{\mathrm{w}}^{-2} I + A_D)^{-1} b\right). \qquad (3.31)
\end{aligned}$$

Diese Abschätzung ist möglich, da beide Supremumsargumente stetig sind und für $\sigma_{\mathrm{w}} \to \infty$ konvergieren, und daher die Suprema existieren. Für $M > 1$ gilt

$$\int_{\sigma_{\mathrm{w}}^{\mathrm{opt}}}^{\infty} \sigma_{\mathrm{w}}^{-M}\, d\sigma_{\mathrm{w}} = \frac{1}{M-1} \left(\sigma_{\mathrm{w}}^{\mathrm{opt}}\right)^{-M+1}, \qquad (3.32)$$

sodass alle Faktoren auf der rechten Seite von Ungleichung 3.31 beschränkt sind. Damit ist gezeigt, dass beide Integrale der Gleichung 3.25 existieren. Wie bereits erwähnt gibt es allerdings zwei Ausnahmen. Falls A_D singulär ist, liegen zu wenige Trainingsdaten vor oder die vorhandenen Trainingsdaten enthalten nach Abbildung durch die Basisfunktionen lineare Abhängigkeiten; dieser Fall wird in Abschnitt 3.2.3 im Zusammenhang mit numerischen Problemen diskutiert. Die zweite Ausnahme $M = 1$ ist für die Praxis nicht relevant. Intuitiv ist es auch wenig sinnvoll für ein einzelnes Gewicht einen regulierenden Hyperparameter einzuführen.

Nun soll ein Verfahren beschrieben werden, das eine Lösung $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ der Gleichung 3.25 berechnet. Die linke Seite der Gleichung 3.25 ist monoton steigend in $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ während die rechte Seite monoton fallend in $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ ist. Diese Beobachtung garantiert nicht nur die Eindeutigkeit der Lösung $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ und die numerische Stabilität bei ihrer Berechnung, sondern führt auch zu folgendem Entwurf des Algorithmus.

Sei $\sigma_{\mathrm{w}}^{(1)} < \ldots < \sigma_{\mathrm{w}}^{(J)}$ eine Menge von Stützstellen, an denen der Integrand $p(\sigma_{\mathrm{w}}|D)$ ausgewertet werden soll. Da die Größe $p(\sigma_{\mathrm{w}}|D)$ so extremen Schwankungen in der Größenordnung unterliegt, dass sie oft nicht mehr als Standardgleitkommazahl (IEEE 754, „double“) dargestellt werden kann, wird in der Implementierung nicht der Ausdruck nach Gleichung 3.27, sondern die Funktion

$$\begin{aligned}
L(\sigma_{\mathrm{w}}^{(j)}) &:= -M \ln \sigma_{\mathrm{w}}^{(j)} - \frac{1}{2} \ln \det((\sigma_{\mathrm{w}}^{(j)})^{-2} I + A_D) + \frac{1}{2} b^T ((\sigma_{\mathrm{w}}^{(j)})^{-2} I + A_D)^{-1} b \\
&= const + \ln p(\sigma_{\mathrm{w}}^{(j)}|D) \qquad (3.33)
\end{aligned}$$

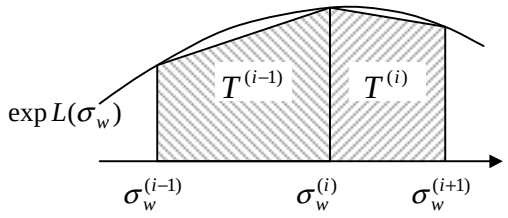

Abbildung 3.5: Approximation der a posteriori Dichte von σ_w durch Trapeze

berechnet. Gleichung 3.25 lautet nun

$$\int_0^{\sigma_\mathrm{w}^\mathrm{opt}} \exp(L(\sigma_\mathrm{w}))\, d\sigma_\mathrm{w} \quad = \quad \int_{\sigma_\mathrm{w}^\mathrm{opt}}^{\infty} \exp(L(\sigma_\mathrm{w}))\, d\sigma_\mathrm{w}. \tag{3.34}$$

Das Integral zwischen zwei Stützstellen $\sigma_\mathrm{w}^{(j)}$ und $\sigma_\mathrm{w}^{(j+1)}$ wird nun durch ein Trapez mit dem Flächeninhalt T_j approximiert (zur Trapezregel siehe Abbildung 3.5 und [Forster]):

$$\begin{aligned} T_j \quad &\propto \quad \int_{\sigma_\mathrm{w}^{(j)}}^{\sigma_\mathrm{w}^{(j+1)}} p(\sigma_\mathrm{w}|D)\, d\sigma_\mathrm{w} \\ T_j \quad &:= \quad c \cdot \frac{1}{2}(\sigma_\mathrm{w}^{(j+1)} - \sigma_\mathrm{w}^{(j)}) \left(\exp\left(L(\sigma_\mathrm{w}^{(j)})\right) + \exp\left(L(\sigma_\mathrm{w}^{(j+1)})\right)\right) \quad \text{für } j = 1, \ldots, J-1 \end{aligned} \tag{3.35}$$

mit einer Konstanten c. Diese Konstante wird nun so gewählt, dass die Argumente der Exponentialfunktionen, die im Algorithmus berechnet werden, in Größenordnungen liegen, die zu keinen Problemen mit der Gleitkommadarstellung führen. Sei daher

$$c \;:=\; \exp(-L_\mathrm{max}) \qquad \text{mit} \qquad L_\mathrm{max} \;:=\; \max_{j=1,\ldots,J} L(\sigma_\mathrm{w}^{(j)}), \tag{3.36}$$

dann gilt

$$T_j \quad = \quad \frac{1}{2}(\sigma_\mathrm{w}^{(j+1)} - \sigma_\mathrm{w}^{(j)}) \left(\exp\left(L(\sigma_\mathrm{w}^{(j)}) - L_\mathrm{max}\right) + \exp\left(L(\sigma_\mathrm{w}^{(j+1)}) - L_\mathrm{max}\right)\right). \tag{3.37}$$

Nun werden Berechnungen der Exponentialfunktion nur noch mit nicht positiven Argumenten durchgeführt. Dabei spielen Rundungsfehler bei sehr kleinen Argumenten keine Rolle mehr, da diese ohnehin kaum einen Integralbeitrag liefern.

Um das Integral auf der linken Seite von Gleichung 3.25 zu approximieren, wird die erste Stützstelle $\sigma_\mathrm{w}^{(1)} = 0$ gewählt. Logarithmiert man Gleichung 3.30, so erhält man direkt

$$L(0) \quad := \quad \lim_{\sigma_\mathrm{w} \to 0} L(\sigma_\mathrm{w}) \;=\; 0, \tag{3.38}$$

somit ist der Integrand im gesamten Intervall $[0, \infty[$ definiert und die erste Stützstelle muss nicht algorithmisch berechnet werden.

Für das letzte „Trapez“ T_J wird aufgrund der unbeschränkten Integrationslänge folgende Festlegung getroffen:

$$\begin{aligned} T_J \quad &\propto \quad \int_{\sigma_\mathrm{w}^{(J)}}^{\infty} p(\sigma_\mathrm{w}|D)\, d\sigma_\mathrm{w} \\ T_J \quad &:= \quad c \cdot \int_{\sigma_\mathrm{w}^{(J)}}^{2\sigma_\mathrm{w}^{(J)}} \exp\left(L(\sigma_\mathrm{w}^{(J)})\right)\, d\sigma_\mathrm{w} \\ &= \quad \sigma_\mathrm{w}^{(J)} \exp\left(L(\sigma_\mathrm{w}^{(J)}) - L_\mathrm{max}\right). \end{aligned} \tag{3.39}$$

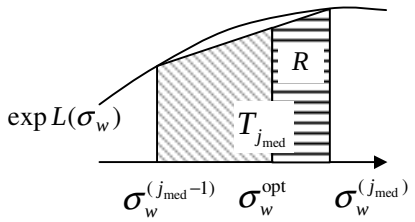

Abbildung 3.6: Bestimmung von $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ im medianen Trapez $T_{j_{\mathrm{med}}}$

Es wird vermutet, dass T_J meistens größer als das zu approximierende Integral ist. Diese Eigenschaft forciert jedoch den Algorithmus (s.u.) dieses letzte Trapez genauer zu untersuchen und verringert so das Risiko einen wesentlichen Integralanteil hinter der letzten Stützstelle $\sigma_{\mathrm{w}}^{(J)}$ zu übersehen.

Nach Berechnung aller Trapeze $T_1, \ldots, T_J$ prüft der Algorithmus, ob die Auswahl der Stützstellen ausreichend fein war und verwendet dazu eine heuristische Regel: der Anteil jedes Trapezes darf höchstens das ϵ-fache des approximierten Gesamtintegrals (linke plus rechte Seite von Gleichung 3.25) betragen. Formal lautet diese Regel

$$\forall j \in \{1, \ldots, J\} \qquad : \qquad T_j \leq \epsilon \sum_{i=0}^{J} T_i. \tag{3.40}$$

Der Wert $\epsilon = 0,05$ hat sich bewährt.

Ist Bedingung 3.40 nicht erfüllt, so wird das größte Trapez $T_{j\max}$ durch eine neue Stützstelle unterteilt. Der Algorithmus geht dazu wie folgt vor: zunächst wird der Index $j_{\max}$ des größten Trapezes bestimmt, bei Nichteindeutigkeit wird ein beliebiger Index ausgewählt, es gilt dann $T_{j\max} \geq T_j$ für alle j. Anschließend wird eine neue Stützstelle

$$\sigma_{\mathrm{w}}^{\mathrm{neu}} \quad := \quad \begin{cases} \frac{1}{2}\left(\sigma_{\mathrm{w}}^{(j\max)} + \sigma_{\mathrm{w}}^{(j\max+1)}\right), & \text{falls } j_{\max} < J \\ 2 \cdot \sigma_{\mathrm{w}}^{(J)} & , \text{falls } j_{\max} = J \end{cases} \tag{3.41}$$

im Inneren des betroffenen Trapezes erzeugt und entsprechend einsortiert.

Falls Bedingung 3.40 erfüllt ist, terminiert die Generierung von Stützstellen; in diesem Fall sind beide Integrale der Gleichung 3.25 genau genug approximiert. Daneben gibt es aber noch eine alternative Terminierungsbedingung, nämlich dann, wenn $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ genau genug bestimmt ist. Die hierzu passende heuristische Bedingung basiert auf der Intervalllänge des größten Trapezes und lautet

$$j_{\max} < J \qquad \text{und} \qquad \sigma_{\mathrm{w}}^{(j\max+1)} - \sigma_{\mathrm{w}}^{(j\max)} \leq \delta \sigma_{\mathrm{w}}^{(j\max)} \tag{3.42}$$

für eine Konstante δ, für die der Wert $0,0001$ gewählt wurde. Diese Terminierungsbedingung spart Rechenzeit, wenn $p(\sigma_{\mathrm{w}}|D)$ ein sehr scharfes Maximum besitzt.

Nachdem eine der beiden Terminierungsbedingungen erfüllt ist, kann $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ direkt bestimmt werden, indem in Gleichung 3.25 die Integrale durch ihre Trapezapproximationen ersetzt werden. Der Integrand ist nun stückweise linear. Zunächst muss bestimmt werden, in welches Trapez $T_{j_{\mathrm{med}}}$ die gesuchte Stelle $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ fällt: der Index j_{med} erfüllt dabei die Bedingung

$$\sum_{j=0}^{j_{\mathrm{med}}-1} T_j < \frac{1}{2} \sum_{j=0}^{J} T_j \leq \sum_{j=0}^{j_{\mathrm{med}}} T_j \tag{3.43}$$

und ist durch sie auch eindeutig bestimmt. Die algorithmische Umsetzung ist einfach.

Der letzte Schritt beinhaltet die exakte Bestimmung von $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ im Trapez $T_{j_{\mathrm{med}}}$ (siehe dazu Abbildung 3.6). Es ergibt sich folgende Lösung:

$$R \quad := \quad \sum_{j=0}^{j_{\mathrm{med}}} T_j - \frac{1}{2} \sum_{j=0}^{J} T_j$$

$$
\begin{aligned}
d &:= \frac{L(\sigma_{\mathrm{w}}^{(j_{\mathrm{med}})}) - L(\sigma_{\mathrm{w}}^{(j_{\mathrm{med}}-1)})}{2\left(\sigma_{\mathrm{w}}^{(j_{\mathrm{med}})}) - \sigma_{\mathrm{w}}^{(j_{\mathrm{med}}-1)}\right)} \\
\sigma_{\mathrm{w}}^{\mathrm{opt}} &= \sigma_{\mathrm{w}}^{(j_{\mathrm{med}})} - \left| \frac{L(\sigma_{\mathrm{w}}^{(j_{\mathrm{med}})})}{2d} \right| - \sqrt{\frac{L(\sigma_{\mathrm{w}}^{(j_{\mathrm{med}})})^2}{4d^2} - \frac{R}{d}} \Bigg|. \qquad (3.44)
\end{aligned}
$$

Damit ist das gesamte Verfahren zur approximativen Bestimmung von $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ durch den Medianansatz beschrieben. Es folgt zusammenfassend eine Skizze des Algorithmus:

```
Variablen:
      Σ = [σ_w^(1); ...; σ_w^(J)]: Folge von Gleitkommazahlen
      L = [L(σ_w^(1)); ...; L(σ_w^(J))]: Folge von Gleitkommazahlen

Wähle initiale Stützstellen: Σ ← [0; 1]
Berechne L(1) (Gleichung 3.33)
Setze L ← [0; L(1)] (Gleichung 3.38)
SCHLEIFE:
      Berechne T_1, ..., T_J (Gleichungen 3.37 und 3.39)
      Falls Bedingung 3.40 erfüllt ist, gehe zu ENDE:
      Bestimme j_max
      Falls Bedingung 3.42 erfüllt ist, gehe zu ENDE:
      Berechne σ_w^neu (Gleichung 3.41)
      Berechne L(σ_w^neu) (Gleichung 3.33)
      Füge σ_w^neu in Σ und L(σ_w^neu) in L ein
      Gehe zu SCHLEIFE:
ENDE:
Bestimme j_med (Gleichung 3.43)
Bestimme σ_w^opt (Gleichung 3.44)
```

3.2.3 Behandlung numerischer Probleme

Während der Berechnung der Stützstellenwerte der Funktion $L(\sigma_{\mathrm{w}})$ treten numerische Probleme bei der Inversion der Matrix A auf: statt A^{-1} wird eine Matrix $B \approx A^{-1}$ berechnet. Da A symmetrisch und positiv definit ist, bietet sich für die Matrixinversion die Cholesky-Zerlegung an, die nicht nur schneller, sondern auch numerisch stabiler als die RL- oder die QR-Zerlegung ist ([Meister], [PreTeuVet]), und die darüber hinaus auch die Symmetrie der Inversen garantiert. Nach der Inversion kann man eine Fehlermatrix $C := AB - I$ berechnen, die idealerweise die Nullmatrix ist, die aber in der Praxis teilweise recht große Komponenten enthalten kann.

Bekanntlich setzt sich $A = \sigma_{\mathrm{w}}^{-2} I + A_D$ aus zwei Teilmatrizen zusammen. Die erste, $\sigma_{\mathrm{w}}^{-2} I$, ist immer positiv definit, kann jedoch für große Werte von σ_{w} betragsmäßig sehr klein gegenüber der zweiten Matrix A_D werden, die nicht von σ_{w} abhängt. Die Matrix A_D ist zwar immer positiv semidefinit, jedoch kann ihr Rang teilweise erheblich unter ihrer Dimension M liegen; die Ursachen dafür sind eine schlechte Wahl von Basisfunktionen oder affin lineare Abhängigkeiten in den Trainingsstellen. Strategien zur Vermeidung von affin linearen Abhängigkeiten werden bei der Vorverarbeitung der Trainingsdaten berücksichtigt und in den Abschnitten 5.4.3 und 5.4.6 diskutiert. Lineare Abhängigkeiten lassen sich aber nicht immer völlig vermeiden, sodass hier eine Diskussion und algorithmische Behandlung von numerischen Fehlern notwendig ist.

Der numerische Fehler bei der Matrixinversion wirkt sich direkt auf den berechneten Wert der Funktion $L(\sigma_{\mathrm{w}})$ aus. Nach Gleichung 3.33 besteht diese aus drei Summanden, von denen aber die ersten beiden im Vergleich zum dritten numerisch stabil sind. Sei daher mit $L_{\mathrm{math}} = b^T A^{-1} b$ der exakte dritte Summand und mit $L_{\mathrm{num}} = b^T B b$ der numerisch ermittelte dritte Summand bezeichnet (der Faktor 1/2 wird hier der Einfachheit halber weggelassen).

Man kann nun den entstandenen Fehler $L_{\text{num}} - L_{\text{math}}$ wie folgt schätzen:

$$\begin{aligned} B - A^{-1} &= B - A^{-1} + A^{-1}(C - AB + I) \\ &= B - A^{-1} + A^{-1}C - B + A^{-1} \\ &= A^{-1}C \qquad (3.45) \\ L_{\text{num}} - L_{\text{math}} &= b^T Bb - b^T A^{-1} b \\ &= b^T A^{-1} Cb \\ &\approx b^T BCb \\ &= w_{\text{MP}}^T Cb \qquad (3.46) \end{aligned}$$

Dieser letzte Term kann prinzipiell berechnet werden. Es hat sich allerdings nicht bewährt ihn zur Korrektur von B zu verwenden, da dann Probleme mit der Symmetrie der Matrizen B und C auftreten.

Man kann aber die Funktion $L(\sigma_{\text{w}})$ um einen negativen Strafsummanden ergänzen, der die Wirkung numerischer Fehler verringert. Man beachte, dass die Integrationsfunktion $\exp L(\sigma_{\text{w}})$ lautet und somit schon kleine Rundungsfehler von $L(\sigma_{\text{w}})$ nach oben extreme Auswirkungen haben können. Der Strafsummand wird wie folgt modelliert: es wird angenommen, dass die Elemente der Matrix C stochastisch unabhängige, normalverteilte Zufallsvariablen mit Erwartungswert Null sind. Der Strafsummand ist dann

$$\begin{aligned} -\sqrt{VAR_C[L_{\text{num}} - L_{\text{math}}]} &\approx -\sqrt{VAR\left[w_{\text{MP}}^T Cb\right]} \\ &= -\sqrt{VAR\left[\sum_{m=1}^{M}\sum_{i=1}^{M}(w_{\text{MP}})_m C_{mi} b_i\right]} \\ &= -\sqrt{\sum_{m=1}^{M}\sum_{i=1}^{M}(w_{\text{MP}})_m^2 b_i^2 VAR[C_{mi}]}. \qquad (3.47) \end{aligned}$$

Er kann durch den Ausdruck

$$L_{\text{straf}} = -\sqrt{\sum_{m=1}^{M}\sum_{i=1}^{M}(w_{\text{MP}})_m^2 b_i^2 C_{mi}^2} \qquad (3.48)$$

geschätzt werden. Im Algorithmus wird also der Summand L_{num} durch die Summanden $L_{\text{num}} + L_{\text{straf}}$ ersetzt.

Abschließend soll hier noch die Wirkung des Strafsummanden diskutiert werden. Er bewirkt zwar keine Korrektur des Integranden von Gleichung 3.25, der mit einem numerischen Rauschen überlagert ist, er führt aber zu einem kleineren Integranden dort, wo das numerische Rauschen groß ist. Dies wiederum führt zu einem kleineren $\sigma_{\text{w}}^{\text{opt}}$, denn das numerische Rauschen wächst mit σ_{w}. Das Netz wird stärker regularisiert und man erkennt dies an einer schlechteren Korrelation zwischen Trainings- und Prognosewerten sowie an verkleinerten Prognosefehlern. Würde man den Strafsummanden aber weglassen, wäre die Berechnung von Prognosewerten und -fehlern numerisch instabil: die Korrelation zwischen Trainings- und Prognosewerten wäre ebenfalls schlecht und der Prognosefehler würde unzuverlässig. Ohne Strafsummand wurden Fälle beobachtet, in denen die Matrix B nicht mehr positiv definit war und daher negative Prognosevarianzen (3.14) berechnet wurden, was eine anschließende Kooperation von Netzen, Abschnitt 4.1, unmöglich machte.

3.2.4 Training mit Hilfe der Eigenwertzerlegung

Bei empirischen Tests mit realen Korrosionsdaten stellte sich die Numerik als Problem dar. Schon bei wenigen hundert Datensätzen war das berechnete $\sigma_{\text{w}}^{\text{opt}}$ aufgrund des numerischen Strafterms spürbar zu klein. Insbesondere wurde die Anzahl der Datensätze pro Netz weniger durch die Rechenzeit, sondern vielmehr durch die Numerik beschränkt.

Während Abschnitt 3.2.3 einen gangbaren, wenn auch nicht zufrieden stellenden Weg darstellt, soll in diesem Abschnitt eine alternative Vorgehensweise beschrieben werden, die auf einer Idee von Prof. Anlauf basiert. Die durch Gleichung 3.29 definierte Matrix A_D ist symmetrisch und lässt sich daher in

$$A_D = U\Lambda U^T \tag{3.49}$$

zerlegen. Dabei ist $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_M)$ die Diagonalmatrix mit den Eigenwerten $\lambda_1 \geq \ldots \geq \lambda_M$ von A_D und $U = (u_1, \ldots, u_M) \in \mathbb{R}^{M\times M}$ die zugehörige Orthonormalmatrix, spaltenweise bestehend aus den Eigenvektoren $u_1, \ldots, u_M$ von A_D. Diese Zerlegung wird nun vom Trainingsalgorithmus berechnet. Die Berechnung geschieht iterativ ([PreTeuVet]) und ist mit $O(M^3)$ rechenintensiv, man beachte aber, dass sie nur einmal während des Netztrainings durchgeführt werden muss.

Es wird davon ausgegangen, dass die Berechnung der Matrizen Λ und U numerisch stabil ist. Da A_D positiv semidefinit ist, müssen alle Eigenwerte nicht negativ sein. Daher werden alle berechneten negativen Eigenwerte auf den Wert 0 gesetzt, um auch nach der Zerlegung die Eigenschaft der positiven Semidefinitheit sicher zu stellen.

Betrachten wir nun die durch Gleichung 3.33 gegebene Funktion $L(\sigma_\mathrm{w})$, die während der iterativen Bestimmung von $\sigma_\mathrm{w}^\mathrm{opt}$ in jeder Iteration an einer Stelle σ_w ausgewertet werden muss.

$$\begin{aligned}
L(\sigma_\mathrm{w}) &= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\ln\det(\sigma_\mathrm{w}^{-2}I + U\Lambda U^T) + \frac{1}{2}b^T(\sigma_\mathrm{w}^{-2}I + U\Lambda U^T)^{-1}b \\
&= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\ln\det(\sigma_\mathrm{w}^{-2}UIU^T + U\Lambda U^T) + \frac{1}{2}b^T(\sigma_\mathrm{w}^{-2}UIU^T + U\Lambda U^T)^{-1}b \\
&= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\ln\det(U(\sigma_\mathrm{w}^{-2}I + \Lambda)U^T) + \frac{1}{2}b^T(U(\sigma_\mathrm{w}^{-2}I + \Lambda)U^T)^{-1}b \\
&= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\ln\det(\sigma_\mathrm{w}^{-2}I + \Lambda) + \frac{1}{2}b^TU(\sigma_\mathrm{w}^{-2}I + \Lambda)^{-1}U^Tb
\end{aligned} \tag{3.50}$$

Mit der Größe $\tilde{b} := U^Tb = (\tilde{b}_1, \ldots, \tilde{b}_M)^T$, die nur einmalig während des Trainings berechnet werden muss, ergibt sich weiter

$$\begin{aligned}
L(\sigma_\mathrm{w}) &= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\ln\det(\sigma_\mathrm{w}^{-2}I + \Lambda) + \frac{1}{2}\tilde{b}^T(\sigma_\mathrm{w}^{-2}I + \Lambda)^{-1}\tilde{b} \\
&= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\ln\det\begin{pmatrix} \sigma_\mathrm{w}^{-2}+\lambda_1 & & \\ & \ddots & \\ & & \sigma_\mathrm{w}^{-2}+\lambda_M \end{pmatrix} \\
&\qquad + \frac{1}{2}\tilde{b}^T\begin{pmatrix} \frac{1}{\sigma_\mathrm{w}^{-2}+\lambda_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_\mathrm{w}^{-2}+\lambda_M} \end{pmatrix}\tilde{b} \\
&= -M\ln\sigma_\mathrm{w} - \frac{1}{2}\sum_{m=1}^{M}\ln(\sigma_\mathrm{w}^{-2}+\lambda_m) + \frac{1}{2}\sum_{m=1}^{M}\frac{\tilde{b}_m^2}{\sigma_\mathrm{w}^{-2}+\lambda_m}.
\end{aligned} \tag{3.51}$$

In dieser Form lässt sich die Funktion L in Zeit $O(M)$ an einer Stelle σ_w auswerten. Die Zeit für einen Iterationsschritt bei der Bestimmung von $\sigma_\mathrm{w}^\mathrm{opt}$ sinkt also von $O(M^3)$ wie in Abschnitt 3.2.2 beschrieben auf M-lineare Zeit, wenn man einmalig in die Eigenwertzerlegung 3.49 investiert.

Nach dem Training könnte nun die für die Prognosefehler benötigte inverse Hesse-Matrix durch den Ausdruck

$$\begin{aligned}
A^{-1} &= U(\sigma_\mathrm{w}^{-2}I + \Lambda)^{-1}U^T \\
&= U\begin{pmatrix} \frac{1}{\sigma_\mathrm{w}^{-2}+\lambda_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_\mathrm{w}^{-2}+\lambda_M} \end{pmatrix}U^T
\end{aligned} \tag{3.52}$$

berechnet werden. Leider ist so aufgrund der Numerik die berechnete Matrix A^{-1} nicht mehr exakt symmetrisch und somit auch nicht mehr positiv definit. Die Folge sind negative berechnete Prognosevarianzen, die weitere Berechnungen wie etwa die Kooperation unmöglich machen. Betrachten wir daher die Prognosevarianz neu:

$$\begin{aligned} \sigma^2(x) &= g(x)^T A^{-1} g(x) \\ &= g(x)^T U(\sigma_{\mathrm{w}}^{-2} I + \Lambda)^{-1} U^T g(x). \end{aligned} \tag{3.53}$$

Bei der Prognose wird nun die Größe $\tilde{g} := U^T g(x) = (\tilde{g}_1, \ldots, \tilde{g}_M)^T$ berechnet und die Prognosevarianz ergibt sich dann zu

$$\begin{aligned} \sigma^2(x) &= \tilde{g}^T(\sigma_{\mathrm{w}}^{-2} I + \Lambda)^{-1} \tilde{g} \\ &= \sum_{m=1}^{M} \frac{\tilde{g}_m^2}{\sigma_{\mathrm{w}}^{-2} + \lambda_m}. \end{aligned} \tag{3.54}$$

Diese Form der Berechnung ist numerisch stabil: die Multiplikation mit $U^T g(x)$ ist stabil, da U als Orthonormalmatrix gut konditioniert ist, und der Term 3.54 ist ebenfalls sehr stabil, da alle Summanden positiv sind.

Die Berechnung der Prognosewerte sollte im Sinne der Konsistenz der Berechnungen ebenfalls über die Orthonormaltransformation durchgeführt werden:

$$\begin{aligned} \mu(x) &= g(x)^T w_{\mathrm{MP}} \\ &= g(x)^T A^{-1} b \\ &= g(x)^T U(\sigma_{\mathrm{w}}^{-2} I + \Lambda)^{-1} U^T b \\ &= \tilde{g}^T(\sigma_{\mathrm{w}}^{-2} I + \Lambda)^{-1} \tilde{b}. \end{aligned} \tag{3.55}$$

Mit der numerisch stabilen Berechnung der Größe $\widetilde{w_{\mathrm{MP}}} = (\sigma_{\mathrm{w}}^{-2} I + \Lambda)^{-1} \tilde{b}$ ergibt sich für den Prognosewert die bekannte Form

$$\mu(x) = \tilde{g}^T \widetilde{w_{\mathrm{MP}}}. \tag{3.56}$$

Die Rechenzeit der Prognose unterscheidet sich asymptotisch von der nach Abschnitt 3.1.1 nur durch eine Konstante. Die Berechnung von $\tilde{g}$ kann in Zeit $O(M^2)$ und die des Ausdrucks 3.54 in Zeit $O(M)$ durchgeführt werden. Die nach Abschnitt 3.1.1 benötigte Speicherplatzgröße ist asymptotisch $\approx M^2/2$, da das speicherplatzbestimmende Objekt die symmetrische Matrix A^{-1} ist. Bei der Eigenwertzerlegung ist das speicherplatzbestimmende Objekt die Orthonormalmatrix U. Man kann sie in $O(M^2)$ Zeit elementweise mit einem Speicherplatzverbrauch von M^2 Gleitkommazahlen speichern, braucht dann also etwa doppelten Speicherplatz. Man beachte, dass in der vorliegenden Implementierung die Speicherplatzgröße direkt mit der Ladezeit der Netze aus der Datenbank korrespondiert.

Eine Orthonormalmatrix kann allerdings auch effizienter gespeichert werden, und zwar mit ebenfalls asymptotisch $\approx M^2/2$ Gleitkommazahlen. Die Idee soll hier kurz beschrieben werden, obwohl eine Implementierung nicht durchgeführt wurde. Grundlage dazu ist das numerische Verfahren der QR-Zerlegung mit Hilfe von Householder-Matrizen, etwa beschrieben in [Meister]. Der Algorithmus bestimmt Householder-Matrizen $H_1, \ldots, H_{M-1} \in \mathbb{R}^{M \times M}$ und eine Rechtsoben-Matrix $R \in \mathbb{R}^{M \times M}$ so, dass $U = H_1 \cdots H_{M-1} R$ gilt. Jede Householder-Matrix H_i ist dabei durch einen Vektor $v_i \in \mathbb{R}^M$ eindeutig gegeben, sodass $H_i = I - 2/(v^T v) \cdot v v^T$ ist. Dabei verschwinden die Elemente $1, \ldots, i-1$ des Vektors v_i; alle Vektoren $v_1, \ldots, v_{M-1}$ zusammen können daher in asymptotisch $\approx M^2/2$ Gleitkommazahlen gespeichert werden.

Die Zerlegung der Matrix $U = QR$, wobei $Q = H_1, \ldots, H_{M-1}$ ist, ist bis auf Multiplikation mit einer Diagonalmatrix $D = \mathrm{diag}(\pm 1, \ldots, \pm 1)$ eindeutig: $U = (QD)(DR)$. Mit $Q = U$ und $R = I$ ist eine Zerlegung gefunden, somit hat die algorithmisch gefundene Dreiecksmatrix die Form von D und kann in nur M Bits gespeichert werden.

Die Householder-Zerlegung muss nach der Eigenwertzerlegung nach Gleichung 3.49 explizit berechnet werden. Dadurch kann zwar gegenüber einer direkten Speicherung von U Speicherplatz gespart werden, es

entstehen jedoch weitere Rundungsfehler. Die Householder-Zerlegung findet nach dem Training statt und ist daher zeitunkritisch, da ihre Rechenzeit die eigentliche Trainingszeit asymptotisch nicht übersteigt. Kritisch ist dagegen die Zeit beim Laden des Netzes: die Multiplikation der Matrizen in der Form

$$U = \left(I - \frac{2}{v_1^T v_1} v_1 v_1^T\right) \cdots \left(I - \frac{2}{v_{M-1}^T v_{M-1}} v_{M-1} v_{M-1}^T\right) D \tag{3.57}$$

würde ebenfalls $O(M^3)$ arithmetische Operationen benötigen und daher asymptotisch länger dauern. Es ist zu untersuchen, ob es hier ein schnelleres Verfahren gibt.

3.3 Wichtige Netzeigenschaften

In den Kapiteln 4 und 5 werden die in diesem Kapitel vorgestellten Netze als Module verwendet, die aus Trainingsdaten Prognosen berechnen. Da das Verhalten der Netze sehr kompliziert ist, werden vereinfachende Näherungen benötigt, um die Netze makroskopisch zu beschreiben. In diesem Abschnitt werden einige derartige Näherungen hergeleitet und erläutert. Außerdem werden weitere wichtige Eigenschaften der Netze beschrieben, die ein besseres Verständnis des Verhaltens der Netze und der Bedingungen und der Güte der Näherungen ermöglichen. Ähnliche und ergänzende Betrachtungen finden sich in [QazWilBis] und [WilQazBis].

3.3.1 Äquivalenz von Messungen an gleicher Stelle

Nehmen wir an, an einer Stelle des Eingangsraums würden mehrere Messungen durchgeführt. Jede dieser Messungen beschreibt eine Verteilung des gesuchten wahren Werts an dieser Stelle. Es ergibt sich daher die Frage, ob es möglich ist, in den Trainingsdaten diese Messungen durch eine einzelne Messung so zu ersetzen, dass das resultierende Netz unverändert bleibt.

Sei dazu $x_1, \ldots, x_N$ eine Menge von Trainingsstellen. Die Stellen $x_1 = \ldots = x_\nu =: x$ seien dabei identisch, über die übrigen Messstellen $x_{\nu+1}, \ldots, x_N$ seien hier keine Annahmen getroffen. Für den Datenteil der Hesse-Matrix des Netzes gilt nun

$$\begin{aligned} A_D &= \sum_{n=1}^{N} s_n^{-2} g(x_n) g(x_n)^T \\ &= \sum_{n=1}^{\nu} s_n^{-2} g(x) g(x)^T + \sum_{n=\nu+1}^{N} s_n^{-2} g(x_n) g(x_n)^T \\ &= \left(\sum_{n=1}^{\nu} s_n^{-2}\right) g(x) g(x)^T + \sum_{n=\nu+1}^{N} s_n^{-2} g(x_n) g(x_n)^T. \end{aligned} \tag{3.58}$$

Setzt man

$$s := \left(\sum_{n=1}^{\nu} s_n^{-2}\right)^{-\frac{1}{2}} \tag{3.59}$$

als Trainingsfehler der ersetzenden Messung, so bleibt A_D unverändert. Betrachten wir ebenso den Vektor

$$\begin{aligned} b &= \sum_{n=1}^{N} t_n s_n^{-2} g(x_n) \\ &= \sum_{n=1}^{\nu} t_n s_n^{-2} g(x) + \sum_{n=\nu+1}^{N} t_n s_n^{-2} g(x_n) \\ &= \left(s^2 \sum_{n=1}^{\nu} t_n s_n^{-2}\right) s^{-2} g(x) + \sum_{n=\nu+1}^{N} t_n s_n^{-2} g(x_n). \end{aligned} \tag{3.60}$$

Setzt man hier

$$t \quad = \quad s^2 \sum_{n=1}^{\nu} t_n s_n^{-2} \tag{3.61}$$

als Trainingswert der ersetzenden Messung, so bleibt auch b unverändert. Durch $A = \sigma_{\mathrm{w}}^{-2} I + A_D$ und b ist $S(w)$ nach Gleichung 3.8 bis auf eine Konstante eindeutig bestimmt. Somit ist die a posteriori Verteilung der Gewichte $p(w|D)$ für jedes σ_{w} eindeutig bestimmt. Daraus folgt, dass durch den Austausch der Trainingsdaten

$$\{(x, t_1, s_1), \ldots, (x, t_\nu, s_\nu)\} \iff \{(x, t, s)\} \tag{3.62}$$

das trainierte Netz nicht verändert wird.

Dieses Resultat ist nicht wirklich verwunderlich. Jede Messung beschreibt über die bayessche Gleichung den gesuchten wahren Wert als Normalverteilung um den Messwert. Die Zusammenfassung der Messungen entspricht dann der Und-Verknüpfung dieser Verteilungen.

3.3.2 Multiplikation der Basisfunktionen

Die Wahl der Basisfunktionen ist wesentlich für die Prognosen. Sie bestimmt die Menge der möglichen Prognosewertfunktionen. Sie bestimmt auch den Grad der Ähnlichkeit verschiedener Stellen und damit den Prognosefehler. Es ist daher notwendig vor der Festlegung der Basisfunktionen das Verhalten der Basisfunktionen unter verschiedenen Transformationen zu betrachten.

Was passiert, wenn alle Basisfunktionen mit einem konstanten Faktor $\alpha \neq 0$ multipliziert werden ? Sei $g(x)$ der alte und $\tilde{g}(x) := \alpha g(x)$ der neue Vektor der Basisfunktionen. Für die aus den Trainingsdaten berechneten neuen Konstanten, die hier ebenso mit Tilden bezeichnet werden, gilt

$$\begin{aligned}
\tilde{A}_D &= \sum_{n=1}^{N} s_n^{-2} \tilde{g}(x_n) \tilde{g}(x_n)^T \\
&= \alpha^2 \sum_{n=1}^{N} s_n^{-2} g(x_n) g(x_n)^T \\
&= \alpha^2 A_D \qquad (3.63) \\
\tilde{b} &= \sum_{n=1}^{N} t_n s_n^{-2} \tilde{g}(x_n) \\
&= \alpha b. \qquad (3.64)
\end{aligned}$$

Daraus ergibt sich die für die Bestimmung von $\widetilde{\sigma_{\mathrm{w}}^{\mathrm{opt}}}$ wichtige Funktion $\tilde{L}(\sigma_{\mathrm{w}})$ nach Gleichung 3.33 zu

$$\begin{aligned}
\tilde{L}(\sigma_{\mathrm{w}}) &= -M \ln \sigma_{\mathrm{w}} - \frac{1}{2} \ln \det(\sigma_{\mathrm{w}}^{-2} I + \tilde{A}_D) + \frac{1}{2} \tilde{b}^T (\sigma_{\mathrm{w}}^{-2} I + \tilde{A}_D)^{-1} \tilde{b} \\
&= -M \ln \sigma_{\mathrm{w}} - \frac{1}{2} \ln \det(\sigma_{\mathrm{w}}^{-2} I + \alpha^2 A_D) + \frac{\alpha^2}{2} b^T (\sigma_{\mathrm{w}}^{-2} I + \alpha^2 A_D)^{-1} b \\
&= -M \ln \sigma_{\mathrm{w}} - \frac{1}{2} \ln (\det(\alpha^{-2} \sigma_{\mathrm{w}}^{-2} I + A_D) \cdot \alpha^{2M}) + \frac{1}{2} b^T (\alpha^{-2} \sigma_{\mathrm{w}}^{-2} I + A_D)^{-1} b \\
&= -M \ln(\alpha \sigma_{\mathrm{w}}) - \frac{1}{2} \ln \det(\alpha^{-2} \sigma_{\mathrm{w}}^{-2} I + A_D) + \frac{1}{2} b^T (\alpha^{-2} \sigma_{\mathrm{w}}^{-2} I + A_D)^{-1} b \\
&= L(\alpha \sigma_{\mathrm{w}}). \qquad (3.65)
\end{aligned}$$

Diese Gleichung legt die Beziehung $\widetilde{\sigma_{\mathrm{w}}^{\mathrm{opt}}} \approx \alpha^{-1} \sigma_{\mathrm{w}}^{\mathrm{opt}}$ nahe, unabhängig von der Bestimmungsmethode von $\sigma_{\mathrm{w}}^{\mathrm{opt}}$. Speziell für die Bestimmung über den Median ergibt sich die gesuchte Gewichtsregularisierung $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ als Lösung der Gleichung 3.34. Wenn $\sigma_{\mathrm{w}}^{\mathrm{opt}}$ Gleichung 3.34 für L erfüllt, dann erfüllt sie auch $\widetilde{\sigma_{\mathrm{w}}^{\mathrm{opt}}} = \alpha^{-1} \sigma_{\mathrm{w}}^{\mathrm{opt}}$

für $\tilde{L}$, wie man durch Anwendung der Substitutionsregel erkennen kann:

$$\begin{aligned}
\int_0^{\sigma_\mathrm{w}^\mathrm{opt}} \exp(L(\sigma_\mathrm{w}))\,d\sigma_\mathrm{w} &= \int_{\sigma_\mathrm{w}^\mathrm{opt}}^{\infty} \exp(L(\sigma_\mathrm{w}))\,d\sigma_\mathrm{w} \\
\int_0^{\sigma_\mathrm{w}^\mathrm{opt}} \exp(\tilde{L}(\alpha^{-1}\sigma_\mathrm{w}))\,d\sigma_\mathrm{w} &= \int_{\sigma_\mathrm{w}^\mathrm{opt}}^{\infty} \exp(\tilde{L}(\alpha^{-1}\sigma_\mathrm{w}))\,d\sigma_\mathrm{w} \\
\int_0^{\alpha^{-1}\sigma_\mathrm{w}^\mathrm{opt}} \exp(\tilde{L}(\sigma_\mathrm{w}))\alpha\,d\sigma_\mathrm{w} &= \int_{\alpha^{-1}\sigma_\mathrm{w}^\mathrm{opt}}^{\infty} \exp(\tilde{L}(\sigma_\mathrm{w}))\alpha\,d\sigma_\mathrm{w} \\
\int_0^{\widetilde{\sigma_\mathrm{w}^\mathrm{opt}}} \exp(\tilde{L}(\sigma_\mathrm{w}))\,d\sigma_\mathrm{w} &= \int_{\widetilde{\sigma_\mathrm{w}^\mathrm{opt}}}^{\infty} \exp(\tilde{L}(\sigma_\mathrm{w}))\,d\sigma_\mathrm{w}. \qquad (3.66)
\end{aligned}$$

Betrachten wir nun die neuen Prognosen:

$$\begin{aligned}
\tilde{\mu}(x) &= \tilde{g}(x)^T \tilde{A}^{-1} \tilde{b} \\
&= \alpha g(x)^T ((\alpha^{-1}\sigma_\mathrm{w}^\mathrm{opt})^{-2} I + \alpha^2 A_D)^{-1} \alpha b \\
&= g(x)^T ((\sigma_\mathrm{w}^\mathrm{opt})^{-2} I + A_D)^{-1} b \\
&= \mu(x), \qquad (3.67) \\
\tilde{\sigma^2}(x) &= \tilde{g}(x)^T \tilde{A}^{-1} \tilde{g}(x) \\
&= \alpha g(x)^T ((\alpha^{-1}\sigma_\mathrm{w}^\mathrm{opt})^{-2} I + \alpha^2 A_D)^{-1} \alpha g(x) \\
&= g(x)^T ((\sigma_\mathrm{w}^\mathrm{opt})^{-2} I + \alpha^2 A_D) g(x) \\
&= \sigma^2(x). \qquad (3.68)
\end{aligned}$$

Die Multiplikation aller Basisfunktionen mit einem konstanten Faktor ändert somit zwar alle internen Größen des Netzes, nicht aber die Prognosen.

3.3.3 Orthonormale Transformation der Basisfunktionen

Werden bei einem Netz die Basisfunktionen linear und orthonormal transformiert, ändert sich sein Verhalten nicht. Dies kann man wie folgt einsehen: sei $U \in \mathbb{R}^{M \times M}$ eine beliebige Orthonormalmatrix ($U^T U = I$), sei weiter $g(x)$ der alte und $\tilde{g}(x) := Ug(x)$ der neue Vektor der Basisfunktionen. Die Größen des alten Netzes werden auch in diesem Abschnitt ohne Tilden und die Größen des neuen Netzes mit Tilden (˜) notiert. Für die neuen Versionen der in Gleichung 3.8 definierten Größen A und b gilt nun

$$\begin{aligned}
\tilde{A} &= \sigma_\mathrm{w}^{-2} I + \sum_{n=1}^{N} s_n^{-2} \tilde{g}(x) \tilde{g}(x)^T \\
&= \sigma_\mathrm{w}^{-2} I + \sum_{n=1}^{N} s_n^{-2} U g(x) g(x)^T U^T \\
&= U \left(\sigma_\mathrm{w}^{-2} I + \sum_{n=1}^{N} s_n^{-2} g(x) g(x)^T \right) U^T \\
&= UAU^T \qquad \text{und} \qquad (3.69) \\
\tilde{b} &= \sum_{n=1}^{N} t_n s_n^{-2} \tilde{g}(x_n) \\
&= U \sum_{n=1}^{N} t_n s_n^{-2} g(x_n) \\
&= Ub. \qquad (3.70)
\end{aligned}$$

Daraus folgt für die Prognosen, die durch die Gleichungen 3.13 und 3.14 gegeben sind

$$\begin{aligned}
\tilde{\mu}(x) &= \tilde{g}(x)^T \tilde{A}^{-1} \tilde{b} \\
&= g(x)^T U^T U A^{-1} U^T U b \\
&= g(x)^T A^{-1} b \\
&= \mu(x) \qquad \text{und} \qquad (3.71) \\
\tilde{\sigma}^2(x) &= \tilde{g}(x)^T \tilde{A}^{-1} \tilde{g}(x) \\
&= g(x)^T U^T U A^{-1} U^T U g(x) \\
&= g(x)^T A^{-1} g(x) \\
&= \sigma^2(x). \qquad (3.72)
\end{aligned}$$

Es kann ebenso leicht gezeigt werden, dass die Funktion $L(\sigma_w)$ nach Gleichung 3.33 unter der Orthonormaltransformation durch U invariant bleibt. Somit ist auch die Bestimmung von σ_w^{opt} invariant.

Die Invarianz bezüglich einer beliebigen orthonormalen Transformation der Basisfunktionen hat natürlich wichtige Auswirkungen auf die Auswahl der Basisfunktionen. Zwei spezielle Fälle einer solchen Transformation sollen hier explizit genannt werden. Die Permutation der Basisfunktionen ist eine orthonormale Transformation und daher eine Invariante; diese Eigenschaft überrascht nicht.

Für den zweiten Fall nehmen wir an, dass die Basisfunktionen $g_1, \ldots, g_M$ linear abhängig seien. Dann existieren Koeffizienten $c_1, \ldots, c_M$, die nicht alle verschwinden, mit der Eigenschaft

$$\sum_{m=1}^{M} c_m g_m = 0. \qquad (3.73)$$

Ohne Beschränkung der Allgemeinheit kann zusätzlich die Normierung

$$\sum_{m=1}^{M} c_m^2 = 1 \qquad (3.74)$$

der Koeffizienten gefordert werden. Schreibt man $c = (c_1, \ldots, c_M)^T$ für den Vektor der Koeffizienten, können die Gleichungen 3.73 und 3.74 als

$$\begin{aligned}
c^T g &= 0 \qquad \text{und} \qquad (3.75) \\
\|c\| &= 1 \qquad (3.76)
\end{aligned}$$

geschrieben werden. Zum gegebenen Vektor c kann man nun Vektoren $u_1, \ldots, u_{M-1}$ finden, sodass die Vektoren $u_1, \ldots, u_{M-1}, c$ eine Orthonormalbasis des $\mathbb{R}^M$ bilden (Basisergänzungssatz und schmidtsches Orthogonalisierungsverfahren, siehe dazu etwa [NieWer]). Die oben verwendete Matrix U wird nun zeilenweise aus diesen Basisvektoren gebildet. Für die neuen Basisfunktionen gilt dann

$$\tilde{g} = Ug = \begin{pmatrix} u_{1,1} & \cdots & u_{1,M} \\ \vdots & & \vdots \\ u_{M-1,1} & \cdots & u_{M-1,M} \\ c_1 & \cdots & c_M \end{pmatrix} \begin{pmatrix} g_1 \\ \vdots \\ g_M \end{pmatrix} = \begin{pmatrix} ? \\ \vdots \\ ? \\ c^T g \end{pmatrix} = \begin{pmatrix} ? \\ \vdots \\ ? \\ 0 \end{pmatrix}. \qquad (3.77)$$

Die mit ? bezeichneten Komponenten des neuen Vektors der Basisfunktionen ergeben sich hier eindeutig, sind aber nicht weiter von Interesse. Wichtig ist, dass die M-te Basisfunktion verschwindet. Man kann also aus einem System von M Basisfunktionen, die linear abhängig sind, eine Basisfunktion so eliminieren, also M um Eins verkleinern, ohne das Verhalten des Netzes zu verändern, indem man die übrigen Basisfunktionen geeignet transformiert. Man beachte, dass eine verschwindende Basisfunktion algorithmisch weggelassen und so Rechenzeit und Speicherplatz gespart werden kann. Diese Erkenntnis ist wichtig bei der konkreten Wahl der Basisfunktionen.

Während des Trainings werden die Basisfunktionen nur an den Messstellen $x_1, \ldots, x_N$ evaluiert. Man könnte daher den Definitionsbereich der Basisfunktionen auf diese Stellen beschränken und die obigen

Aussagen wären uneingeschränkt gültig. Eine Reduktion der Anzahl der Basisfunktionen wäre demnach bereits durch eine lineare Abhängigkeit der Basisfunktionen an den höchstens N verschiedenen Messstellen möglich.

Zur Veranschaulichung dieses Gedankens betrachten wir folgendes Beispiel: über einer einzelnen Eingangsvariablen x bilden die Basisfunktionen

$$g(x) = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix} \tag{3.78}$$

eine Basis des Vektorraums der Polynome von maximalem Grad 2. Wurden Messungen an nur zwei verschiedenen Stellen $x_1 = 0$ und $x_2 = 1$ durchgeführt, existiert dort die lineare Abhängigkeit $g_2(x) = g_3(x)$ für $x \in \{x_1, x_2\}$. Dies rechtfertigt aber keine Reduktion der Basisfunktionen auf nur zwei, wenn Prognosen für beliebige $x \in \mathbb{R}$ zu erwarten sind[3].

Dagegen beinhaltet die Wahl

$$g(x) = \begin{pmatrix} 1 \\ x \\ 2x-1 \end{pmatrix} \tag{3.79}$$

eine lineare Abhängigkeit, die zu einer Reduktion der Basisfunktionen führen sollte. Die Reduktion selbst ist kompliziert und hier durch die Wahl der $u_1, \ldots, u_N$ und Gleichung 3.77 gegeben. Es ist nicht bekannt, ob es eine einfachere Methode der Reduktion gibt, das direkte Weglassen einer Basisfunktion ist allerdings nicht möglich. Ist für ein m der Koeffizient $c_m \neq 0$, dann kann die sich für einen bestimmten Gewichtsvektor w ergebende Netzfunktion $g(x, w) = g(x)^T w$ zwar durch das Weglassen der m-ten Basisfunktion auch durch neue, direkt reduzierte Vektoren $g'(x)^T w'$ dargestellt werden, der Gewichtsvektor w' hat dann aber eine andere a priori Wahrscheinlichkeitsdichte $p(w')$. Bildet man also ein neues Netz durch direktes Weglassen einer Basisfunktion, so sind altes und neues Netz in der Regel nicht äquivalent.

3.3.4 Lineare Transformation der Basisfunktionen

Nachdem gezeigt wurde, dass die orthonormale Transformation der Basisfunktionen das Verhalten des Netzes nicht verändert, stellt sich die Frage nach einer allgemeineren linearen Transformation. Sei V eine invertierbare $M \times M$-Matrix, die den Vektor der Basisfunktionen g in einen Vektor neuer Basisfunktionen $\tilde{g} = Vg$ transformiert. Für die neuen Netzgrößen gilt dann ähnlich den Gleichungen 3.69 und 3.70

$$\begin{aligned} \tilde{A}_D &= V A_D V^T, && (3.80) \\ \tilde{A}^{-1} &= \left(\sigma_w^{-2} I + V A_D V^T\right)^{-1} \\ &= \left(V \left(\sigma_w^{-2} (V^T V)^{-1} + A_D\right) V^T\right)^{-1} \\ &= (V^T)^{-1} \left(\sigma_w^{-2} (V^T V)^{-1} + A_D\right)^{-1} V^{-1} \qquad \text{und} && (3.81) \\ \tilde{b} &= Vb. && (3.82) \end{aligned}$$

Daraus folgt für die Prognosen:

$$\begin{aligned} \tilde{\mu}(x) &= \tilde{g}(x)^T \tilde{A}^{-1} \tilde{b} \\ &= g(x)^T V^T (V^T)^{-1} \left(\sigma_w^{-2} (V^T V)^{-1} + A_D\right)^{-1} V^{-1} V b \\ &= g(x)^T \left(\sigma_w^{-2} (V^T V)^{-1} + A_D\right)^{-1} b \qquad \text{und} && (3.83) \\ \tilde{\sigma}^2(x) &= \tilde{g}(x)^T \tilde{A}^{-1} \tilde{g}(x) \\ &= g(x)^T V^T (V^T)^{-1} \left(\sigma_w^{-2} (V^T V)^{-1} + A_D\right)^{-1} V^{-1} V g(x) \\ &= g(x)^T \left(\sigma_w^{-2} (V^T V)^{-1} + A_D\right)^{-1} g(x). && (3.84) \end{aligned}$$

[3] Anders sieht es aus, wenn lineare Abhängigkeiten durch das Problem bzw. das Datenmodell vorgegeben sind und diese Abhängigkeiten sowohl für die Trainingsstellen als auch für mögliche Prognosestellen gelten. In diesem Fall ist ebenfalls eine entsprechende Reduktion günstig.

Diese Prognosen unterscheiden sich von denen des ursprünglichen Netzes nur in der Matrix $\sigma_{\mathrm{w}}^{-2}(V^TV)^{-1}$, die ursprünglich $\sigma_{\mathrm{w}}^{-2}I$ lautete. Verfolgt man den Ursprung dieses Terms anhand der Gleichungen aus Abschnitt 3.1 zurück, so beschreibt er die a priori Verteilung der Gewichte. Das neue Netz ist demnach äquivalent zu einem Netz mit den alten Basisfunktionen g und folgender veränderten a priori Verteilung der Gewichte:

$$w \quad \propto \quad \mathcal{N}\left(0, \sigma_{\mathrm{w}}^2 V^TV\right). \tag{3.85}$$

Man beachte, dass die Matrix V^TV immer symmetrisch und positiv definit ist, sie stellt hier die Kovarianzmatrix der Gewichtsverteilung dar. Die neuen Gewichte sind zwar noch normalverteilt, jetzt aber stochastisch abhängig.

Die Äquivalenz der Transformation durch V steht in engem Zusammenhang mit Transformationen nach Abschnitt 2.7.

3.3.5 Zusammenhang zwischen den Trainingsdaten und den Prognosen

In diesem Abschnitt werden vereinfachende Näherungen für das Verhalten der Netze hergeleitet, die in den Abschnitten 4.1 (Kooperation), 4.3 (diskontinuierliche Ausgänge) und 4.4 (regionales Rauschen) verwendet werden.

Betrachten wir eine Prognose an einer Stelle x, an der auch eine Messung vorliegt: $x = x_n$ für ein $n \in \{1, \ldots, N\}$. Nehmen wir an, dass diese Stelle x_n weit weg von den anderen Messstellen liegt. „Weit weg“ bedeutet in diesem Zusammenhang, dass die Messung bei x_n keinen Einfluss auf Prognosen an den anderen Messstellen hat. Die gewählte mathematische Formulierung dieser Annahme lautet: die durch die a priori Verteilung der Gewichte w induzierte Zufallsvariable $g(x_n, w)$ ist stochastisch unabhängig von jeder der ebenso induzierten Zufallsvariablen $g(x_i, w)$ für $i \neq n$.

Die Gewichte sind a priori $\mathcal{N}\left(0, \sigma_{\mathrm{w}}^2 I\right)$-verteilt. Daraus folgt

$$\begin{aligned}
\begin{pmatrix} g(x_n, w) \\ g(x_i, w) \end{pmatrix} &= \begin{pmatrix} g(x_n)^T w \\ g(x_i)^T w \end{pmatrix} \\
&= \begin{pmatrix} g_1(x_n) & \cdots & g_M(x_n) \\ g_1(x_i) & \cdots & g_M(x_i) \end{pmatrix} w \\
&\propto \mathcal{N}\left(0, \begin{pmatrix} g_1(x_n) & \cdots & g_M(x_n) \\ g_1(x_i) & \cdots & g_M(x_i) \end{pmatrix} \sigma_{\mathrm{w}}^2 I \begin{pmatrix} g_1(x_n) & g_1(x_i) \\ \vdots & \vdots \\ g_M(x_n) & g_M(x_i) \end{pmatrix}\right) \\
&= \mathcal{N}\left(0, \sigma_{\mathrm{w}}^2 \begin{pmatrix} \sum_{m=1}^M g_m(x_n)^2 & \sum_{m=1}^M g_m(x_n) g_m(x_i) \\ \sum_{m=1}^M g_m(x_i) g_m(x_n) & \sum_{m=1}^M g_m(x_i)^2 \end{pmatrix}\right) \\
&= \mathcal{N}\left(0, \sigma_{\mathrm{w}}^2 \begin{pmatrix} \|g(x_n)\|^2 & g(x_n)^T g(x_i) \\ g(x_n)^T g(x_i) & \|g(x_i)\|^2 \end{pmatrix}\right)
\end{aligned} \tag{3.86}$$

für jedes $i \neq n$. Man sieht, dass die normalverteilten Zufallsvariablen $g(x_n, w)$ und $g(x_i, w)$ genau dann stochastisch unabhängig sind, wenn sie unkorreliert sind, was wiederum genau dann der Fall ist, wenn der Ausdruck $g(x_n)^T g(x_i)$ verschwindet. Wir gehen im weiteren Verlauf davon aus, dass $g(x_n) \neq 0$ ist, da ansonsten alle Prognosen an der Stelle x_n immer verschwinden würden, $\mu(x_n) = 0$ und $\sigma(x_n) = 0$, was in praktischen Anwendungen wenig sinnvoll erscheint.

Man kann nun die Basisfunktionen so transformieren, dass eine von ihnen an allen Stellen x_i mit $i \neq n$ verschwindet. Sei dazu $u_1 := \|g(x_n)\|^{-1} \cdot g(x_n)$. Gemäß Basisergänzungssatz und schmidtschem Orthogonalisierungsverfahren können nun weitere Vektoren $u_2, \ldots, u_M$ gefunden werden, sodass $\{u_1, \ldots, u_M\}$ eine Orthonormalbasis des $\mathbb{R}^M$ ist. Gemäß Abschnitt 3.3.3 lassen sich nun die Basisfunktionen g durch die Basisfunktionen $\tilde{g} := Ug$ unter Invarianz des Netzverhaltens ersetzen, wobei die Matrix U zeilenweise aus den Vektoren $u_1, \ldots, u_M$ besteht. Es gilt für die neuen Basisfunktionen

$$\tilde{g}_1(x) \quad = \quad u_1^T g(x)$$

$$= \frac{1}{\|g(x_n)\|} g(x_n)^T g(x) \tag{3.87}$$

$$\tilde{g}_1(x_i) = 0 \qquad \text{für } i \neq n \tag{3.88}$$

$$\tilde{g}_1(x_n) = \|g(x_n)\| > 0 \tag{3.89}$$

$$\tilde{g}_m(x_n) = u_m^T g(x_n) = 0 \qquad \text{für } m = 2, \ldots, M \tag{3.90}$$

und zusammenfassend

$$\tilde{g}(x_n) = \begin{pmatrix} \|g(x_n)\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.91}$$

$$\tilde{g}(x_i) = \begin{pmatrix} 0 \\ ? \\ \vdots \\ ? \end{pmatrix} \qquad \text{für } i \neq n, \tag{3.92}$$

wobei mit ? Elemente bezeichnet sind, die zwar eindeutig bestimmt sind, deren Wert aber für die weiteren Berechnungen irrelevant ist.

Betrachten wir nun die internen Netzgrößen

$$\tilde{A} = \sigma_{\mathrm{w}}^{-2} I + \sum_{i=1}^{N} s_i^{-2} \tilde{g}(x_i) \tilde{g}(x_i)^T$$

$$= \begin{pmatrix} \sigma_{\mathrm{w}}^{-2} + s_n^{-2}\|g(x_n)\|^2 & 0 & \cdots & 0 \\ 0 & ? & \cdots & ? \\ \vdots & \vdots & & \vdots \\ 0 & ? & \cdots & ? \end{pmatrix} \tag{3.93}$$

$$\tilde{A}^{-1} = \begin{pmatrix} \frac{1}{\sigma_{\mathrm{w}}^{-2} + s_n^{-2}\|g(x_n)\|^2} & 0 & \cdots & 0 \\ 0 & ? & \cdots & ? \\ \vdots & \vdots & & \vdots \\ 0 & ? & \cdots & ? \end{pmatrix} \tag{3.94}$$

$$\tilde{b} = \sum_{i=1}^{N} t_i s_i^{-2} \tilde{g}(x_i)$$

$$= \begin{pmatrix} t_n s_n^{-2} \|g(x_n)\| \\ ? \\ \vdots \\ ? \end{pmatrix}. \tag{3.95}$$

Die Netzprognosen an der Trainingsstelle x_n ergeben sich nun daraus zu

$$\mu(x_n) = \tilde{g}(x_n)^T \tilde{A}^{-1} \tilde{b}$$

$$= \|g(x_n)\| \cdot \frac{1}{\sigma_{\mathrm{w}}^{-2} + s_n^{-2}\|g(x_n)\|^2} \cdot t_n s_n^{-2} \|g(x_n)\|$$

$$= \frac{t_n}{\sigma_{\mathrm{w}}^{-2} s_n^2 \|g(x_n)\|^{-2} + 1} \qquad \text{und} \tag{3.96}$$

$$\sigma^2(x_n) = \tilde{g}(x_n)^T \tilde{A}^{-1} \tilde{g}(x_n)$$

$$= \|g(x_n)\| \cdot \frac{1}{\sigma_{\mathrm{w}}^{-2} + s_n^{-2}\|g(x_n)\|^2} \cdot \|g(x_n)\|$$

$$= \frac{s_n^2}{\sigma_{\mathrm{w}}^{-2} s_n^2 \|g(x_n)\|^{-2} + 1}. \tag{3.97}$$

Der Prognosewert ist offensichtlich leider nicht erwartungstreu, d.h. der Prognosewert entspricht nicht dem Trainingswert. Er ist in Richtung 0 verschoben, da der Nenner der Gleichung 3.96 echt größer als 1 ist. Gleiches gilt auch für die Prognosevarianz: auch hier ist der gleiche Nenner der Gleichung 3.97 echt größer als 1 und verkleinert daher die Prognosevarianz. Das Netz ist sich in seiner Prognose also sicherer als die Messung. Dies ist auch die Aussage von Lemma 2 in Anhang B auf Seite 166.

Die Ursache für diese Fehlprognosen liegt in der Gewichtsregularisierung. Wir treffen nun eine zweite Annahme: die Gewichtsregularisierung ist vernachlässigbar. Dies entspricht einer a priori Verteilung der Gewichte mit sehr großer Varianz, wir betrachten also den Grenzwert $\sigma_{\mathrm{w}} \to \infty$.

$$\lim_{\sigma_{\mathrm{w}} \to \infty} \mu(x_n) = t_n \tag{3.98}$$

$$\lim_{\sigma_{\mathrm{w}} \to \infty} \sigma^2(x_n) = s_n^2 \tag{3.99}$$

Diese Vorgehensweise ist gerechtfertigt, da das a priori Wissen generell sehr wenig informativ sein sollte, um viele mögliche Prognosefunktionen zuzulassen. Empirisch kann dies bestätigt werden.

Nach Abschnitt 3.3.1 kann nun die eine Messung an der Stelle x_n unter Invarianz des trainierten Netzes durch mehrere Messungen an dieser Stelle ersetzt werden. Wir approximieren hier ebenfalls, indem wir auch Stellen aus der nahen Umgebung zulassen. Zusammenfassend gilt daher:

Seien $(t_1, s_1), \ldots, (t_N, s_N)$ Messwerte und -fehler an Messstellen nahe der Stelle x. Ein Netz, das mit diesen Messungen und weiteren Messungen, deren Stellen weit weg von x liegen, trainiert wurde, berechnet an der Stelle x die Prognose

$$\mu(x) \approx \left(\sum_{n=1}^{N} s_n^{-2} \right)^{-1} \sum_{n=1}^{N} t_n s_n^{-2} \tag{3.100}$$

$$\sigma^2(x) \approx \left(\sum_{n=1}^{N} s_n^{-2} \right)^{-1} . \tag{3.101}$$

Die Näherungen 3.100 und 3.101 bilden eine wichtige Grundlage für die in Kapitel 5 erläuterten Modelle. Sie werden dort in Form von (exakten) Gleichungen verwendet, über die das Verhalten der Netze beschrieben wird. Man beachte, dass die Näherungen 3.100 und 3.101 sehr einfach strukturiert sind und nur von den wenigen Variablen der Messungen in der Nähe von x Gebrauch machen; sie sind insbesondere nicht von σ_{w} abhängig, dessen Wert sich aus der Gesamtheit der Messungen ergibt und daher global ist.

Die Herleitung der Gleichungen 3.100 und 3.101 ist zwar auf generalisierte lineare Netze beschränkt, es wird aber vermutet, dass andere Netztypen diesen Gleichungen ebenfalls (approximativ) genügen. Daher werden sie hier als universell für alle Netztypen gültig angenommen. Dies ist wichtig, wenn die Modelle in Kapitel 5 mit alternativen zugrunde liegenden Netztypen angewendet werden sollen. Die Realisierung und Implementierung der neuronalen Netze ist somit austauschbar und unabhängig von den übrigen Konzepten. [WilQazBis] stellt ergänzende Überlegungen an.

3.3.6 Zusammenhang zwischen den Basisfunktionen und dem Prognosefehler

Auch an Stellen, die „weit weg“ von allen Messstellen liegen, berechnet das Netz endliche Prognosewerte und -fehler. Diese werden durch die a priori Verteilung der Gewichte induziert.

Betrachten wir dazu ein Netz ohne Trainingsdaten ($N = 0$) und fester Gewichtsregularisierung σ_{w}. Es gilt

$$A = \sigma_{\mathrm{w}}^{-2} I \tag{3.102}$$

$$b = 0 \tag{3.103}$$

$$\mu(x) = g(x)^T A^{-1} b$$

$$= 0 \tag{3.104}$$

$$\sigma^2(x) = g(x)^T A^{-1} g(x)$$

$$= \sigma_w^2 \|g(x)\|^2. \tag{3.105}$$

Dass die a priori Prognose des Netzes einen verschwindenden Erwartungswert hat, sollte bei einer Transformation der Ausgangsgröße berücksichtigt werden. Betrachten wir als Beispiel die Prognose der Größe *morgiger Luftdruck in mBar* im Rahmen einer Wettervorhersage. Der athmosphärische Luftdruck hat (in Meereshöhe) einen mittleren Wert von 1013,25mBar, schwankt aber nur um wenige 10mBar. Würde der Luftdruck also direkt als Ausgangsgröße verwendet, so würden abseits der Messdaten Drücke um 0 und — wie in Abschnitt 3.3.5 gezeigt wurde — selbst in der Nähe der Trainingsdaten zu kleine Drücke prognostiziert. Daher ist es angebracht, statt des Luftdrucks selbst die Größe *morgiger Luftdruck in mBar minus 1013,25* als Ausgangsgröße zu verwenden. Diese neue Größe hat nun den (a priori) mittleren Wert 0 und wird weniger starken Verzerrungen durch die Gewichtsregularisierung unterliegen[4].

Gleichung 3.105 zeigt, dass der a priori Prognosefehler nicht nur von der a priori Verteilung der Gewichte in Form von σ_w abhängt, sondern auch sehr wesentlich von den Basisfunktionen. Bei den meisten Problemen unterliegen unter Abwesenheit von Messungen alle Stellen der gleichen Unsicherheit. Es ist daher bei der Wahl der Basisfunktionen anzuraten, den Ausdruck $\|g(x)\|$ für alle potenziellen Prognosestellen x etwa konstant zu halten. Insbesondere sollte es keine Stelle x geben, an der alle Basisfunktionen verschwinden, $g(x) = 0$. An einer solchen Stelle wäre nicht nur der Prognosewert bei beliebigen Trainingsdaten immer 0, sondern auch der Prognosefehler würde immer verschwinden, das Netz wäre sich seiner Prognose also immer sehr sicher.

Es gibt nur sehr wenige Probleme, bei denen bereits a priori wesentliche Eigenschaften der Prognosefunktion bekannt sind. Man muss sich bei der Wahl der Basisfunktionen stets vergegenwärtigen, dass die Menge der möglichen Netzfunktionen gerade der Spann der Basisfunktionen ist. Daher sollten sich die gemeinsamen Eigenschaften der Basisfunktionen auf diejenigen beschränken, die auch das zugrunde liegende Problem aufweist; die Wahl der Basisfunktionen entspricht in diesem Sinne a priori Wissen über das Problem. Hat man kein a priori Wissen über das Problem, so sollten die Basisfunktionen gerade keine gemeinsamen Eigenschaften haben. So sollten etwa Symmetrieen oder Translationsformen immer nur einen Teil der Basisfunktionen betreffen. Die einzige sichere Eigenschaft der Korrosion ist die Stetigkeit in allen Eingangsparametern.

3.3.7 Einflüsse der Eingänge auf die Prognosen

Wie stark beeinflusst eine Messung an einer Stelle x_1 die Prognosen in ihrer Umgebung, wie groß ist diese Umgebung ? In den vorangegangenen Abschnitten wurde hier stark abstrahiert: Messungen an nahe beieinander liegenden Stellen wurden als Messungen an derselben Stelle approximiert, die übrigen Messungen wurden als völlig unabhängig voneinander angenommen. Tatsächlich beeinflusst natürlich jede Messung die Prognosen im gesamten Eingangsraum mehr oder weniger.

Wir betrachten in diesem Abschnitt eine einzelne Messung ($N = 1$) und ihre Wirkung auf den Prognosefehler. Dieser ist

$$\sigma^2(x) = g(x)^T \left(\sigma_w^{-2} I + s_1^{-2} g(x_1) g(x_1)^T\right)^{-1} g(x). \tag{3.106}$$

Ohne Beschränkung der Allgemeinheit können wir nach den Abschnitten 3.3.3 und 3.3.2 annehmen, dass die Basisfunktionen an der Stelle x_1 den ersten Einheitsvektor abbilden: $g(x_1) = (1, 0, \ldots, 0)^T$. Somit folgt:

$$\sigma^2(x) = g(x)^T \begin{pmatrix} (\sigma_w^{-2} + s_1^{-2}) & & & \\ & \sigma_w^{-2} & & \\ & & \ddots & \\ & & & \sigma_w^{-2} \end{pmatrix}^{-1} g(x)$$

[4] Es drängt sich an dieser Stelle die Frage auf, welchen Einfluss eine Multiplikation der Ausgangsgröße hat. Es lässt sich aber leicht zeigen, dass diese Transformationsart eine Invariante darstellt.

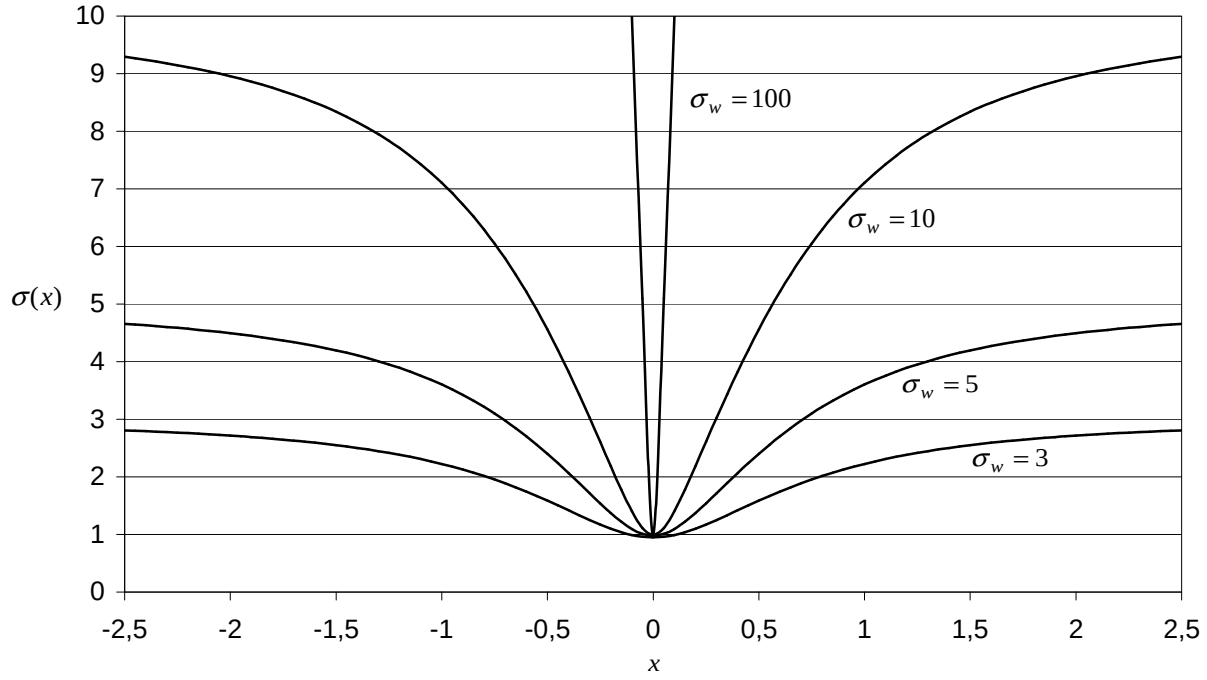

Abbildung 3.7: Wirkung der Gewichtsregularisierung σ_w auf den Prognosefehler in der Umgebung einer einzelnen Messung.

$$
\begin{aligned}
&= g(x)^T \begin{pmatrix} \frac{1}{\sigma_\mathrm{w}^{-2}+s_1^{-2}} & & & \\ & \sigma_\mathrm{w}^2 & & \\ & & \ddots & \\ & & & \sigma_\mathrm{w}^2 \end{pmatrix} g(x) \\
&= \frac{g_1(x)^2}{\sigma_\mathrm{w}^{-2}+s_1^{-2}} + \sum_{m=2}^{M} \sigma_\mathrm{w}^2 g_m(x)^2 \\
&= \frac{g_1(x)^2}{\sigma_\mathrm{w}^{-2}+s_1^{-2}} - \sigma_\mathrm{w}^2 g_1(x)^2 + \sum_{m=1}^{M} \sigma_\mathrm{w}^2 g_m(x)^2 \\
&= \frac{-\sigma_\mathrm{w}^2 s_1^{-2} g_1(x)^2}{\sigma_\mathrm{w}^{-2}+s_1^{-2}} + \sigma_\mathrm{w}^2 \|g(x)\|^2 \\
&= \sigma_\mathrm{w}^2 \left(\|g(x)\|^2 - g_1(x)^2 \frac{s_1^{-2}}{\sigma_\mathrm{w}^{-2}+s_1^{-2}} \right). \qquad (3.107)
\end{aligned}
$$

Im Inneren der Klammer stehen hier zwei Terme, die verschiedene Komponenten des Prognosefehlers beschreiben. Der Minuend beschreibt den durch die a priori Verteilung der Gewichte induzierten Prognosefehler. Er ist vergleichsweise groß, da die a priori Standardabweichung der Gewichte σ_w in der Regel groß ist. An der Messstelle ist $\|g(x_1)\| = 1$ wie oben festgelegt. Da es keinen Grund gibt, bestimmte Stellen a priori genauer zu prognostizieren als andere, gehen wir im Folgenden davon aus, dass im gesamten Bereich möglicher Prognosestellen x die Basisfunktionen in etwa normiert sind: $\|g(x)\| \approx 1$. Dies stellt auch eine Forderung bei der Wahl der Basisfunktionen dar.

Der Subtrahend in der Klammer von Gleichung 3.107 beschreibt die Verringerung des a priori Prognosefehlers aufgrund von Wissen durch die Messung. Unter der Nebenbedingung $\|g(x)\| = 1$ wird $\sigma^2(x)$ genau dann minimal, wenn $g(x) = g(x_1)$ ist, wenn also bei injektiven Basisfunktionen $x = x_1$ ist. Nach Lemma 2, Anhang B, nimmt $\sigma^2(x_1)$ einen Wert kleiner als s_1 an.

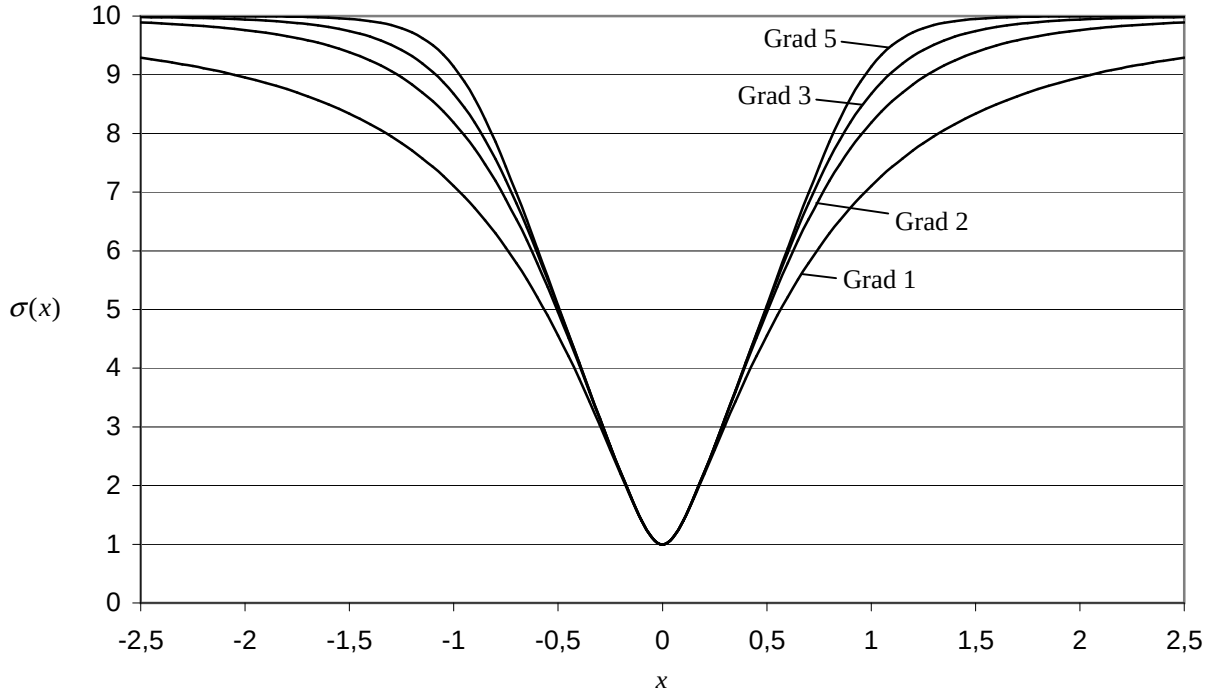

Abbildung 3.8: Wirkung der Anzahl der Basisfunktionen auf den Prognosefehler in der Umgebung. Die Gewichtsregularisierung betrug $\sigma_{\mathrm{w}} = 10$.

Um Gleichung 3.107 weiter zu veranschaulichen verwenden wir im Folgenden ein konkretes Beispiel. Sei $x \in \mathbb{R}$ der einzige Netzeingang und $x_1 = 0$ die einzige Messstelle mit Messfehler $s_1 = 1$. Die Basisfunktionen seien durch

$$\begin{pmatrix} g_1(x) \\ g_2(x) \end{pmatrix} = \frac{1}{\sqrt{1+x^2}} \begin{pmatrix} 1 \\ x \end{pmatrix} \tag{3.108}$$

gegeben. Mit dieser Wahl sind die oben gemachten Annahmen $\|g(x)\| = 1$ und $g(x_1) = (1,0)^T$ erfüllt. Abbildung 3.7 zeigt für dieses Beispiel den Prognosefehler in Abhängigkeit der Prognosestelle x. Definiert man die Umgebung einer Messstelle als das Intervall der Eingangsvariablen x, in dem der Prognosefehler unter einem bestimmten Schwellwert liegt, z.B. $\sigma(x) \leq 2s_1$, so sieht man, dass die Breite der Umgebung sehr stark von der Gewichtsregularisierung abhängt. Bei starker Regularisierung (σ_{w} klein) ist die Umgebung der Messstelle groß, bei schwacher Regularisierung ist sie klein. Durch die Einstellung von σ_{w} kann das Netz also seine Generalisierungseigenschaften bestimmen.

Die Gewichtsregularisierung ist allerdings nicht alleine für die Größe der Umgebung verantwortlich. Eine weitere wichtige Einflussgröße sind natürlich die Basisfunktionen. Verwendet man Basisfunktionen mit kompaktem Träger, so ist die Umgebung natürlich durch diesen Träger beschränkt. Bei allen Basisfunktionen ist die Größe der Umgebung aber auch noch durch andere Faktoren bestimmt. In unserem Beispiel betrachten wir nun verschiedene Basisfunktionen der Art

$$g_m(x) = \frac{1}{\sqrt{\sum_{i=1}^{M} (x^{i-1})^2}} \cdot x^{m-1} \qquad \text{für} \qquad m = 1, \ldots, M. \tag{3.109}$$

Es handelt sich um Monome bis zum Grad $M-1$, die auf die Bedingung $\|g(x)\| = 1$ normiert wurden. Abbildung 3.8 zeigt die Abhängigkeit des Prognosefehlers vom Grad bzw. der Anzahl der Basisfunktionen. Mehr Basisfunktionen führen anscheinend bei konstanter Gewichtsregularisierung zu kleineren Umgebungen.

Die Gewichtsregularisierung und die Anzahl der Basisfunktionen sind zwei wesentliche Einflussgrößen für die Größe der Umgebungen von Messstellen. Zumindest die Gewichtsregularisierung wird jedoch automatisch beim Training bestimmt und ist daher rückgekoppelt an die Anzahl der Basisfunktionen. Es ist daher zu hoffen, dass sich die Umgebungsgröße von Messstellen und daher ein dem Problem adäquates Abstandsmaß im Eingangsraum automatisch auf natürliche Weise einstellt.

3.3.8 Modelle für Abstandsmaße im Eingangsraum

Wann sind zwei Messstellen im Sinne eines Netzes nahe beieinander ? Gibt es eine durch das Netzverhalten induzierte Norm im Eingangsraum $\mathbb{R}^L$? Die Antwort auf diese Fragen ist wichtig, um eine gute Vorverarbeitung von realen Daten durchführen zu können.

Um eine Norm im Eingangsraum finden zu können, muss zunächst festgelegt werden, woran der Abstand zweier Messstellen x_a und x_b phänomenologisch erkennbar sein soll. Dazu werden hier drei Modelle diskutiert. Dem ersten Modell wird die Differenz δ der Prognosewerte

$$\delta \quad = \quad (\mu(x_a) - \mu(x_b))^2 \tag{3.110}$$

zugrunde gelegt. Mit Hilfe diverser Gleichungen und Verteilungen aus Abschnitt 3.1 kann dieser Ausdruck umgeformt werden.

$$\begin{aligned}
\delta &= \left(g(x_a)^T w_{\mathrm{MP}} - g(x_b)^T w_{\mathrm{MP}}\right)^2 \\
&= \left((g(x_a) - g(x_b))^T w_{\mathrm{MP}}\right)^2 \\
&= \left((g(x_a) - g(x_b))^T A^{-1} b\right)^2 \\
&= (g(x_a) - g(x_b))^T A^{-1} b b^T A^{-1} (g(x_a) - g(x_b))
\end{aligned} \tag{3.111}$$

Die Matrix $A^{-1}bb^TA^{-1}$ ist zwar symmetrisch und positiv semidefinit, hat aber höchstens den Rang 1, induziert also durch Gleichung 3.111 noch keine Norm im Gewichtsraum, die in dieser Form zu einer Norm im Eingangsraum weiterentwickelt werden könnte. Sie enthält allerdings den Vektor b, der von den Trainingswerten abhängt, die wiederum von der wahren Funktion abhängen. Ein Abstandsmaß im Eingangsraum sollte aber unabhängig von der wahren Funktion, also a priori, angebbar sein. Wir modellieren daher die wahre Funktion als Zufallsvariable so, dass ihre Werte an den Messstellen $f(x_n) \propto \mathcal{N}\left(0, cs_n^2\right)$-verteilt mit einer Konstanten $c \geq 0$ sind, und die Zufallsvariablen $f(x_1), \ldots, f(x_N)$ paarweise stochastisch unabhängig sind. Für den Fall $c = 0$ ergibt sich $f \equiv 0$, der Fall $c > 0$ setzt voraus, dass alle Messstellen $x_1, \ldots, x_N$ paarweise verschieden sind. Fügt man die Verteilung der wahren Funktion und das Messrauschen zusammen, ergibt sich, dass der n-te Messwert nun $t_n \propto \mathcal{N}\left(0, cs_n^2 + s_n^2\right)$-verteilt ist, und es folgt

$$\begin{aligned}
E[bb^T] &= E\left[\sum_{n=1}^{N} \frac{t_n}{s_n^2} g(x_n) \cdot \sum_{n=1}^{N} \frac{t_n}{s_n^2} g(x_n)^T\right] \\
&= E\left[\sum_{n=1}^{N}\sum_{i=1}^{N} \frac{t_n t_i}{s_n^2 s_i^2} g(x_n) g(x_i)^T\right] \\
&= E\left[\sum_{n=1}^{N} \frac{t_n^2}{s_n^4} g(x_n) g(x_n)^T\right] \\
&= \sum_{n=1}^{N} \frac{c+1}{s_n^2} g(x_n) g(x_n)^T \\
&= (c+1) A_D.
\end{aligned} \tag{3.112}$$

Unter Vernachlässigung der Gewichtsregularisierung ist $E[bb^T] \approx (c+1)A$ und der Erwartungswert für den Abstand kann in Näherung bestimmt werden:

$$\begin{aligned}
E[\delta] &= E\left[(g(x_a) - g(x_b))^T A^{-1} b b^T A^{-1} (g(x_a) - g(x_b))\right] \\
&= (g(x_a) - g(x_b))^T A^{-1} ((c+1) A_D) A^{-1} (g(x_a) - g(x_b)) \\
&\approx (c+1)(g(x_a) - g(x_b))^T A^{-1} (g(x_a) - g(x_b)).
\end{aligned} \tag{3.113}$$

Die Matrix A^{-1} definiert nun durch diese Gleichung eine Norm im Gewichtsraum. Durch eine Taylor-Entwicklung ersten Grads der Basisfunktionen an einer Stelle x_0 in der Nähe der Stellen x_a und x_b wird diese Norm nun auf eine Norm im Eingangsraum abgebildet. Dabei gilt mit $g_0 := g(x_0)$ und der Jacobi-Matrix $D := \nabla_x g(x_0)$ die Gleichung $g(x) = g_0 + D(x - x_0) + o(\|x - x_0\|)$, die als Näherung verwendet wird:

$$\begin{aligned} E[\delta] &\approx (c+1)(g_0 + D(x_a - x_0) - g_0 - D(x_b - x_0))^T A^{-1}(g_0 + D(x_a - x_0) - g_0 - D(x_b - x_0)) \\ &= (c+1)(x_a - x_b)^T D^T A^{-1} D(x_a - x_b). \end{aligned} \tag{3.114}$$

Die Matrix $D^T A^{-1} D$ ist immer symmetrisch und positiv semidefinit. Sie hat in der Regel vollen Rang und definiert dann durch Gleichung 3.114 eine Norm im Eingangsraum, die aber noch von den konkreten Basisfunktionen, den Messstellen und den Messfehlern abhängt. Um auch davon noch zu abstrahieren, werden die Differenziale der Basisfunktionen als Zufallsvariablen angenommen. Für die Elemente der Matrix $D = (d_{ml})_{m=1,\ldots,M;\, l=1,\ldots,L}$ gilt $d_{ml} = (\partial g_m / \partial x_l)(x_0)$, sie werden daher als stochastisch unabhängige Zufallsvariablen $d_{ml} \propto \mathcal{N}\left(0, \delta^2\right)$ für ein $\delta \in \mathbb{R}^+$ modelliert. Die identische Verteilung aller Elemente von D ist innerhalb einer Spalte durch die Gleichrangigkeit der Basisfunktionen untereinander (Permutationsinvarianz), und innerhalb einer Zeile durch die Gleichrangigkeit der Eingänge für jede der Basisfunktionen gerechtfertigt. Die stochastische Abhängigkeit zwischen der Matrix A, die von den Basisfunktionen g direkt abhängt, und der Zufallsmatrix D wird vernachlässigt. Seien $(a_{mi})_{m,i=1,\ldots,M} = A^{-1}$ die Elemente der Inversen der Hesse-Matrix, dann folgt unter Ausdehnung des Erwartungswerts auf die Elemente von D mit diesen Modellannahmen:

$$\begin{aligned} E[\delta] &\approx E\left[(c+1)(x_a - x_b)^T D^T A^{-1} D(x_a - x_b)\right] \\ &= (c+1) E\left[\sum_{l=1}^{L}\sum_{m=1}^{M}\sum_{i=1}^{M}\sum_{j=1}^{L}(x_a - x_b)_l d_{ml} a_{mi} d_{ij} (x_a - x_b)_j\right] \\ &= (c+1) E\left[\sum_{l=1}^{L}\sum_{m=1}^{M}(x_a - x_b)_l d_{ml} a_{mm} d_{ml} (x_a - x_b)_l\right] \\ &= (c+1)\sum_{l=1}^{L}\sum_{m=1}^{M}(x_a - x_b)_l^2 \delta^2 a_{mm} \\ &= (c+1)\delta^2 \mathrm{tr}(A^{-1})\|x_a - x_b\|^2. \end{aligned} \tag{3.115}$$

Somit ist (approximativ) nach all diesen Abstraktionen die im Eingangsraum induzierte Norm die euklidische.

Das zweite Modell legt ebenfalls die Differenz der Prognosewerte nach Gleichung 3.110 zugrunde, abstrahiert aber von vorneherein von den Trainingsdaten. Vielmehr wird der a posteriori wahrscheinlichste Gewichtsvektor als Zufallsvariable betrachtet, der wie die a priori Verteilung der Gewichte verteilt ist: $w_{\mathrm{MP}} \propto \mathcal{N}\left(0, \sigma_{\mathrm{w}}^{-2} I\right)$. Es folgt

$$\begin{aligned} E[\delta] &= E\left[\left((g(x_a) - g(x_b))^T w_{\mathrm{MP}}\right)^2\right] \\ &= E\left[(g(x_a) - g(x_b))^T w_{\mathrm{MP}} w_{\mathrm{MP}}^T (g(x_a) - g(x_b))\right] \\ &= (g(x_a) - g(x_b))^T (\sigma_{\mathrm{w}}^{-2} I)(g(x_a) - g(x_b)). \end{aligned} \tag{3.116}$$

Auch hier werden die Basisfunktionen wieder nach Taylor linear entwickelt und es folgt

$$g(x_a) - g(x_b) = g_0 + D(x_a - x_0) - g_0 - D(x_b - x_0)$$

$$= D(x_a - x_b) \tag{3.117}$$

$$E[\delta] \approx \sigma_{\mathrm{w}}^{-2}(x_a - x_b)^T D^T D(x_a - x_b). \tag{3.118}$$

Die Matrix $D^T D$ induziert hier offensichtlich eine Norm im Eingangsraum, falls sie vollen Rang besitzt, was in der Regel der Fall ist. Die Basisfunktionen sollten so gewählt werden, dass jeder Eingang gleich

stark und die Eingänge unabhängig voneinander berücksichtigt werden. Betrachten wir als Maß dafür die partielle Ableitung der Basisfunktionen $g(x_0)$ nach dem l-ten Eingang, $\partial g/\partial x_l(x_0)$, also die l-te Spalte der Matrix D. Die Basisfunktionen sollten nun so gewählt sein, dass sie die Gleichung

$$\begin{aligned}\left(\frac{\partial g}{\partial x_l}(x_0)\right)^T \frac{\partial g}{\partial x_i}(x_0) &= \sum_{m=1}^{M} d_{ml} d_{mi} \\ &\approx \begin{cases} C & \text{falls } l = i \\ 0 & \text{sonst} \end{cases}\end{aligned} \tag{3.119}$$

für $l, i = 1, \ldots, L$ und ein $C > 0$ in etwa einhalten. Diese Forderung entspricht der Orthogonalität der Spalten in D sowie der euklidischen Normierung auf den Wert $\sqrt{C}$. Da es in der Regel erheblich mehr Basisfunktionen als Eingänge gibt, also mehr Zeilen als Spalten in D, sollte es leicht fallen, die Basisfunktionen entsprechend zu wählen[5]. Dann folgt

$$\begin{aligned}E[\delta] &\approx \sigma_{\mathrm{w}}^{-2} \sum_{l=1}^{L} \sum_{m=1}^{M} \sum_{i=1}^{L} (x_a - x_b)_l d_{ml} d_{mi} (x_a - x_b)_i \\ &\approx \sigma_{\mathrm{w}}^{-2} \sum_{l=1}^{L} \sum_{m=1}^{M} (x_a - x_b)_l d_{ml} d_{ml} (x_a - x_b)_l \\ &\approx \sigma_{\mathrm{w}}^{-2} \sum_{l=1}^{L} (x_a - x_b)_l C (x_a - x_b)_l \\ &= \sigma_{\mathrm{w}}^{-2} C \|x_a - x_b\|^2,\end{aligned} \tag{3.120}$$

womit auch dieses Modell die euklidische Norm im Eingangsraum induziert.

Das dritte Modell betrachtet die Netzausgangsvariable $t|D, \sigma_{\mathrm{w}}, x$ nach Ausdruck 3.12, genauer ihre Kovarianz an den Stellen x_a und x_b. Der Kürze halber sei $t_a := t|D, \sigma_{\mathrm{w}}, x_a$ und $t_b := t|D, \sigma_{\mathrm{w}}, x_b$, dann gilt

$$\begin{aligned}COV[t_a, t_b] &= E\left[(t_a - E[t_a])(t_b - E[t_b])\right] \\ &= E\left[(t_a - \mu(x_a))(t_b - \mu(x_b))\right] \\ &= E\left[(g(x_a)^T w - g(x_a)^T w_{\mathrm{MP}})(g(x_b)^T w - g(x_b)^T w_{\mathrm{MP}})\right] \\ &= E\left[g(x_a)^T (w - w_{\mathrm{MP}})(w - w_{\mathrm{MP}})^T g(x_b)\right] \\ &= g(x_a)^T A^{-1} g(x_b).\end{aligned} \tag{3.121}$$

Die Matrix A^{-1} definiert hier ein Skalarprodukt im Gewichtsraum. Aus ihm kann direkt eine Norm abgeleitet werden:

$$\begin{aligned}COV[t_a, t_b] &= \frac{1}{2}\left(g(x_a)^T A^{-1} g(x_a) + g(x_b)^T A^{-1} g(x_b) - (g(x_a) - g(x_b))^T A^{-1} (g(x_a) - g(x_b))\right) \\ &= \frac{1}{2}\left(\sigma^2(x_a) + \sigma^2(x_b) - (g(x_a) - g(x_b))^T A^{-1} (g(x_a) - g(x_b))\right).\end{aligned} \tag{3.122}$$

Die beiden linken Summanden hängen nur von je einem der beiden verglichenen Stellen x_a und x_b ab und sind daher für die Suche nach einer Norm im Eingangsraum uninteressant. Der Subtrahend dagegen ist für das Abstandsmaß entscheidend, er findet sich auch in genau dieser Form bereits beim ersten Modell in Gleichung 3.113. Durch Einsetzen erhält man

$$2COV[t_a, t_b] \approx \sigma^2(x_a) + \sigma^2(x_b) - E[\delta]. \tag{3.123}$$

Die weitere Diskussion ist nun die gleiche wie beim ersten Modell.

[5] Natürlich hängt D vom Entwicklungspunkt x_0 ab. Gemeint ist hier, dass die Relation 3.119 für jedes x_0 in Näherung eingehalten wird.

Die aufgeführten drei Modelle legen nahe, dass die Ähnlichkeit von zwei Netzeingangsvektoren von ihrem euklidischen Abstand abhängt. Dies war vor den Betrachtungen in diesem Abschnitt nicht klar, denn prinzipiell wäre jede Norm, beispielsweise die Manhattan- oder die Max-Norm oder auch eine in den Eingängen asymmetrische Norm, in Frage gekommen. Allerdings wurden recht weitgehende Annahmen, insbesondere über die Basisfunktionen, getroffen, die letztlich zu diesem Resultat geführt haben.

Die Frage nach einem Abstandsmaß im Eingangsraum ist wichtig für ein gutes Verständnis des Netzverhaltens und — wie bereits erwähnt — für die Konstruktion einer Vorverarbeitung von realen Daten. Die in den vorgestellten Modellen enthaltenen Annahmen über die Basisfunktionen legen im Umkehrschluss Bedingungen nahe, die eingehalten werden sollten, um ein definiertes euklidisches Abstandsmaß im Eingangsraum zu erzeugen. Diese finden im folgenden Abschnitt Anwendung.

3.4 Implementierung

3.4.1 Wahl der Basisfunktionen

Um die Basisfunktionen konkret wählen zu können, muss dem Netz bekannt sein, in welchem Bereich die Trainingsstellen liegen. Zwar ist die Festlegung der Basisfunktionen eine a priori Information und darf daher nach Bayes nicht einfach in Abhängigkeit der Trainingsdaten gewählt werden, jedoch kann die Menge der benötigten Basisfunktionen und damit die Anzahl der Gewichte klein gehalten werden, wenn die prinzipielle Lage der Messstellen berücksichtigt wird.

Der sogenannte Netzbereich ist der kleinste achsenparallele Quader im Eingangsraum, dessen Mittelpunkt der Ursprung ist, und der alle Trainingsstellen enthält. Für den l-ten Netzeingang existiert daher eine Konstante ξ_l, sodass alle Messstellen x_{nl} für den l-ten Eingang die Bedingung

$$x_{nl} \quad \in \quad [-\xi_l, \xi_l] \tag{3.124}$$

für alle $n = 1, \ldots, N$ erfüllen. Der Bereich des Netzes ist durch den Quader

$$Q \; := \; \bigotimes_{l=1}^{L} [-\xi_l, \xi_l] \quad \subset \quad \mathbb{R}^L \tag{3.125}$$

gegeben.

Die Konstanten ξ_l sind für jeden Eingang l individuell, da einzelne Eingänge unterschiedlich stark bei den Messungen variiert werden können. Durch Vorverarbeitungsschritte (Abschnitt 5.4) werden die Eingangsparameter so skaliert, dass die Trainingsstellen das volle Intervall $[-\xi_l, \xi_l]$ auch ausfüllen, d.h.

$$\min_{n \in \{1,\ldots,N\}} x_{nl} \quad = \quad -\xi_l \qquad \text{und} \tag{3.126}$$

$$\max_{n \in \{1,\ldots,N\}} x_{nl} \quad = \quad \xi_l. \tag{3.127}$$

Die Basisfunktionen müssen nun so gewählt werden, dass sie im gesamten $\mathbb{R}^L$ bestimmte Eigenschaften erfüllen. Das Netz soll eine möglichst gute Generalisierungsfähigkeit im Inneren seines Bereichs Q und eine möglichst gute Extrapolationsfähigkeit außerhalb seines Bereichs besitzen.

Es werden nun eine Reihe von Forderungen an die Basisfunktionen gestellt.

1. Die Lage der Basisfunktionen darf nicht von den Trainingsdaten abhängen, weder von x_n, t_n oder s_n (Ausnahme Q). Die Basisfunktionen stellen eine a priori Information dar und dürfen daher nicht etwa zu den Trainingsstellen „passend“ gewählt werden. Einige Algorithmen, die mit RBF-Netzen arbeiten (siehe dazu etwa [Zell]), legen etwa die Zentren der Basisfunktionen in die Nähe der Trainingsstellen. Dies führt im bayesschen Kontext aber zu unrealistisch kleinen Prognosefehlern an Stellen, die abseits der Trainingsdaten liegen.

2. Eine Basisfunktion sollte konstant sein (Bias) und für jeden Eingang sollte eine Basisfunktion direkt proportional zu diesem Eingang sein (lineare Basisfunktionen). Durch diese Basisfunktionen ist dann das einfache, aber universelle Modell der affin linearen Regression im Netzmodell enthalten.

Man kann dieses Teilmodell auch noch a priori hervorheben, indem entweder die a priori Standardabweichung der mit diesen Basisfunktionen assoziierten Gewichte gering gewählt wird oder diese Basisfunktionen mit einer großen Konstanten multipliziert werden.

3. Die Menge aller Basisfunktionen sollte linear unabhängig sein, um Redundanzen in der Netzfunktion zu vermeiden (siehe Abschnitt 3.3.3). Dies sollte auch noch gelten, wenn sich die lineare Unabhängigkeit allein auf die Trainingsstellen beschränkt, und wenn diese stark geclustert liegen. Daher sind beispielsweise stückweise Polynome geringer Ordnung (Splines) ungünstig, da einzelne Stücke in der Regel viele Trainingsstellen überdecken werden.

4. Alle Basisfunktionen sollten einmal stetig differenzierbar sein. Diese Forderung ist in der Beobachtung begründet, dass das zugrunde liegende Phänomen ebenfalls diese Eigenschaft hat. Sie ermöglicht aber auch die effiziente Verwendung der Netzprognosen in nachgeschalteten Verarbeitungsschritten, wie etwa eine Optimierung in den Eingangsparametern.

5. Alle Basisfunktionen sollten einen sehr einfachen Graphen besitzen, monotone Funktionen oder Funktionen mit kompaktem Träger etwa sind denkbar. Komplexe Basisfunktionen, etwa Funktionen mit mehreren lokalen Extrema, Wende- oder Sattelpunkten, können unerwünschte Prognosewert- und -fehlerverläufe verursachen.

6. Der Vektor aller Basisfunktionen sollte im Inneren des Netzbereichs in etwa normiert sein, es sollte $\|g(x)\| \approx const$ für $x \in Q$ gelten, siehe dazu Abschnitt 3.3.7. Außerdem sollten die Basisfunktionen den Netzbereich Q gleichmäßig abdecken, d.h. auch die Zahlen

$$\left(\frac{\partial^i}{\partial x_l^i} g_1(x)\right)^2 + \ldots + \left(\frac{\partial^i}{\partial x_l^i} g_M(x)\right)^2 \tag{3.128}$$

sollten für jedes $i = 0, 1, \ldots$ und jedes $l = 1, \ldots, L$ nicht sehr stark für verschiedene $x \in Q$ schwanken. Bei RBF-Netzen kann diese Eigenschaft etwa durch eine Gleichverteilung der Zentren der Basisfunktionen über den Netzbereich Q erreicht werden, bei sigmoiden Funktionen (siehe unten) durch eine Gleichverteilung der Bezugspunkte und -orientierungen.
Bei einer ungleichmäßigen Abdeckung des Netzbereichs durch die Basisfunktionen kann es passieren, dass einige dicht abgedeckte Regionen trotz vieler dichter Trainingsdaten hohe Prognosefehler zwischen ihren Stellen liefern, während das Netz in anderen, dünn abgedeckten Regionen bei sehr wenigen Trainingsdaten scheinbar hervorragend generalisieren kann.

7. Das Verhalten des Prognosefehlers außerhalb des Netzbereichs hängt von der Verwendung des Netzes ab. Für sein asymptotisches Verhalten bei wachsendem Abstand der Prognosestelle x vom Netzbereich Q soll Folgendes gelten:

   - Bei Netzen für kontinuierliche Parameter sollte der Prognosefehler (etwa linear) wachsen. Dies kann etwa durch asymptotisch linear divergierende Basisfunktionen erreicht werden.
   - Netze zur Prognose des regionalen Rauschens (siehe Abschnitt 4.4.3) sollten idealerweise asymptotisch verschwindende Prognosewerte und wachsende Prognosefehler aufweisen. Beides zusammen ist jedoch nicht möglich, denn wenn man die Prognosewerte bereits a priori durch die Basisfunktionen als Verteilung mit geringer Standardabweichung modelliert, kann ihre a posteriori Standardabweichung in Form des Prognosefehlers nicht größer werden. In der Praxis ist folgender Kompromiss denkbar: wählt man die Basisfunktionen so, dass sie asymptotisch gegen nicht verschwindende Konstanten konvergieren, so werden sowohl der Prognosewert als auch der -fehler gegen eine Konstante konvergieren.
   - Für das asymptotische Verhalten von Netzen für diskontinuierliche Prognosen (Abschnitt 4.3.2) wurden noch keine Forderungen aufgestellt. Daher werden derzeit in der Implementierung die gleichen Basisfunktionen verwendet wie bei den Netzen für die kontinuierlichen Parameter.

8. Die deutlich nicht-linearen Bereiche jeder Basisfunktion sollten möglichst nicht außerhalb des Netzbereichs liegen, da sie dort nicht mehr durch Trainingsdaten erfasst werden. Das Volumen dieser

nicht-linearen Bereiche innerhalb von Q sollte aber nicht verschwindend klein gegenüber dem Volumen von Q sein. Diese Eigenschaft garantiert, dass Q überall durch mehrere lokal nicht-lineare Basisfunktionen abgedeckt wird. Folgt man der Argumentation in Abschnitt 3.3.7, so verhindert dies eine künstliche Einengung des Wirkungsbereichs einzelner Messungen durch die Basisfunktionen; die Größe des Wirkungsbereichs kann und sollte vom Netz im Wesentlichen automatisch durch die Gewichtsregularisierung bestimmt werden.

9. Es sollte nicht mehrere voneinander getrennte, deutlich nicht-lineare Bereiche in einer Basisfunktionen geben, damit der regionale Charakter der Basisfunktionen gewährleistet ist. Insbesondere sollte es keine Symmetrieen geben, die in mehreren Basisfunktionen vorkommen, da sich ansonsten trainierte Daten direkt auf die entsprechenden Symmetriestellen auswirken könnten.

10. Die Menge der Basisfunktionen muss skalierbar sein. Dies bedeutet, dass eine Implementierung zu jeder gewünschten Anzahl eine entsprechende Menge von Basisfunktionen erzeugen kann. Auch bei sehr wenigen Basisfunktionen müssen alle Netzeingänge gleichrangig behandelt werden.

Diese Forderungen sind nicht alle gleichzeitig ohne Kompromisse erfüllbar. Für die praktische Implementierung wurde daher das folgende Verfahren entwickelt.

Die erste Basisfunktion ist immer der Bias (Konstante 1), die darauf folgenden L Basisfunktionen sind die Eingangsvariablen $x_1, \ldots, x_L$. Alle weiteren Basisfunktionen sind nicht-linear und werden zufällig und gleichmäßig über den Netzbereich verteilt gewählt.

Diese Vorgehensweise impliziert, dass die Anzahl der Basisfunktionen immer mindestens $L + 1$ sein muss. Tatsächlich ist es wenig sinnvoll, nur allein affin lineare Basisfunktionen zu verwenden, weshalb festgelegt wurde, dass die Mindestanzahl der nicht-linearen Basisfunktionen mindestens $L/2$ sein muss. Es mag vor dem Hintergrund anderer Anwendungen neuronaler Netze seltsam erscheinen, dass hier über derartige Grenzen diskutiert werden muss. Die konkrete Anwendung in der Korrosion führte aber dazu, dass beispielsweise 20 Trainingsdatensätze mit 30 variierten Eingängen in einem Netz zusammengefasst werden mussten. Natürlich kann hier kaum mit einer echten Generalisierung gerechnet werden, man ist mit einer einfachen „Reproduktion“ der Trainingsdaten bei den Prognosen zufrieden. Aber auch diese erfordert eine adäquate Anzahl von Basisfunktionen.

Die Gesamtanzahl der Basisfunktionen M hängt ansonsten nur noch von der Anzahl der Trainingsdaten N ab und ist, wie die Ergebnisse in Abschnitt 3.4.2 nahelegen, ihr gleich. Bei sehr vielen Trainingsdaten allerdings werden etwa weniger Basisfunktionen verwendet, weil dort inhaltliche Redundanzen in den Trainingsdaten erhofft werden, und um die Prognosezeit nicht allzu groß werden zu lassen. Die genaue Zahl der Basisfunktionen ist durch die Heuristik

$$M \quad = \quad \max\left\{ \left\lceil \frac{3}{2}L + 1 \right\rceil, \left\{ \begin{array}{ll} N, & \text{falls } N \leq 200 \\ N/2 + 100, & \text{sonst} \end{array} \right\} \right\} \tag{3.129}$$

gegeben.

Festzulegen sind nun noch die nicht-linearen Basisfunktionen, die bei den meisten Netzen die überwiegende Mehrheit bilden. Diese haben die Form

$$g_m(x) \quad = \quad f_{akt}\left(d_m^T(x - c_m)\right) \tag{3.130}$$

mit Zufallsvektoren $c_m, d_m \in \mathbb{R}^L$ für $m = L + 2, \ldots, M$. Dabei sind

$$c_m \quad \in \quad Q \tag{3.131}$$

$$d_m \quad \in \quad \left\{ d \in \mathbb{R}^L \;:\; \|d\| = \delta^{-1} \right\} \tag{3.132}$$

jeweils gleichverteilt mit

$$\delta \quad := \quad \frac{1}{2}\sqrt{\sum_{l=1}^{L} \xi_l^2}. \tag{3.133}$$

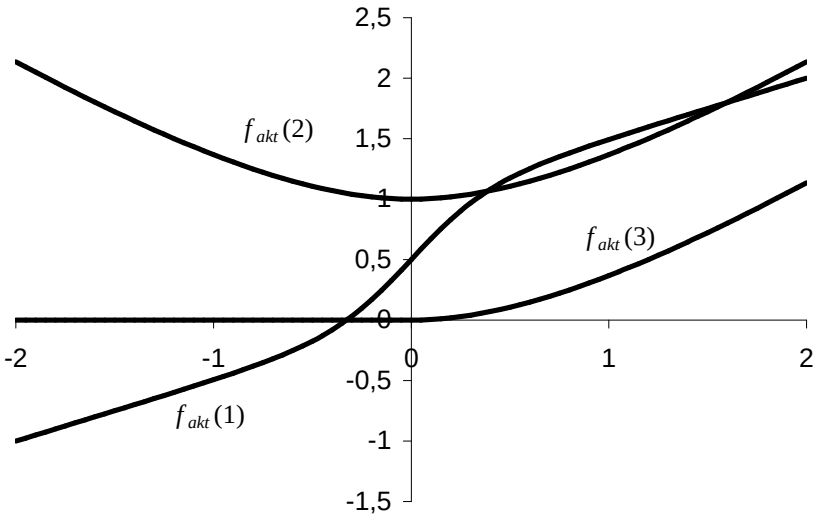

Abbildung 3.9: Graphische Darstellung der Aktivierungsfunktionen nach den Gleichungen 3.135 bis 3.137

Die Konstante δ ist ein Viertel der Diagonallänge von Q und dient als Maßstab für den „aktiven“, nicht-linearen Bereich der Aktivierungsfunktion f_{akt}. Werden nämlich alle Eingänge, also die Eingangsstellen x_{nl} und die Bereiche ξ_l, mit einem konstanten Faktor multipliziert, so bleiben die Werte der Basisfunktionen invariant. Dies wiederum führt zu einem Ähnlichkeitsmaß in den Netzeingängen, das nur auf den relativen Abständen zwischen den einzelnen Eingängen, nicht aber auf absoluten Abständen basiert.

Die Vektoren c_m und d_m legen die m-te Basisfunktion eindeutig fest, besitzen aber redundante Information, denn Gleichung 3.130 kann einfacher mit dem Vektor d_m und dem Skalar $d_m^T c_m$ als

$$g_m(x) \quad = \quad f_{akt}\left(d_m^T x - d_m^T c_m\right) \tag{3.134}$$

geschrieben werden. Es sind daher eigentlich nur $L+1$ Zufallsvariablen (unter der Nebenbedingung $\|d_m\| = \delta^{-1}$) zur Beschreibung einer Basisfunktion notwendig. In der Implementierung ist es aber einfacher die Zufallsvariablen nach den Ausdrücken 3.131 und 3.132 zu erzeugen.

Da die Argumente der Aktivierungsfunktion f_{akt} durch δ normiert sind, ist die Aktivierungsfunktion so zu wählen, dass ihr „aktiver“, nicht-linearer Bereich etwa im Intervall $[-1, 1]$ liegt. Es existieren drei konkrete Vorschläge

$$f_{akt}^{(1)}(x) \quad = \quad \frac{x}{2} + \frac{1}{1+\exp(-5x)} \tag{3.135}$$

$$f_{akt}^{(2)}(x) \quad = \quad |x| + \exp(-|x|) \tag{3.136}$$

$$f_{akt}^{(3)}(x) \quad = \quad \begin{cases} 0 & : \quad x \leq 0 \\ x - 1 + \exp(-x) & : \quad x > 0 \end{cases}, \tag{3.137}$$

die in Abbildung 3.9 dargestellt sind. Derzeit wird $f_{akt}^{(3)}$ verwendet, da sie gegenüber den anderen beiden den Vorteil hat, dass an einer konkreten Stelle x im Mittel die Hälfte aller nicht-linearen Basisfunktionen verschwindet. Die Ausnutzung dieser Tatsache kann zur beschleunigten Berechnung einer Prognose genutzt werden.

3.4.2 Empirische Auswertungen

Die gesamte in diesem Kapitel entwickelte Theorie muss natürlich empirisch mit realen Daten validiert werden. Dazu wurden Daten aus einem gut untersuchten Bereich der Korrosion gewählt: es handelt sich um Schwefelsäure-Medien an verschiedenen Eisenbasisstählen.

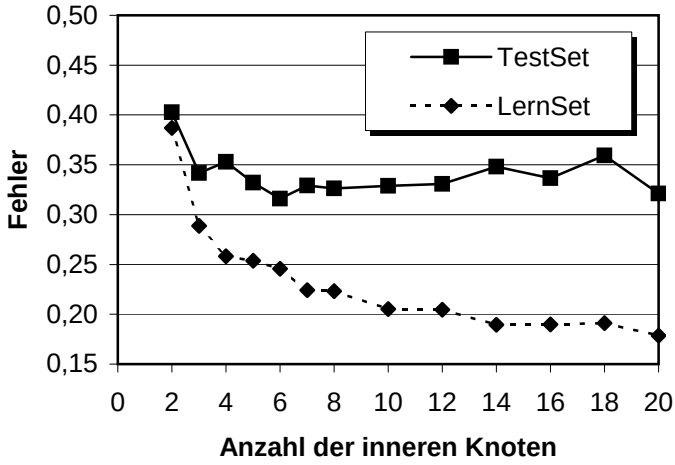

Abbildung 3.10: Mittlerer quadratischer Fehler bei DataModel

Die Datensätze besitzen 12 Eingänge und füllen den Eingangsraum einigermaßen gleichmäßig aus, sind also nicht stark geclustert. Es gibt nur eine Ausgangsvariable. Um die hier beschriebenen Methoden mit klassischen Methoden vergleichen zu können, wurden die Trainingsmessfehler konstant gewählt.

Die insgesamt 533 Datensätze wurden zufällig in ein Lernset mit 267 Datensätzen und ein Testset mit 266 Datensätzen aufgeteilt. Natürlich arbeiten die bayesschen Methoden ohne Testset, auf dem Testset soll jedoch die Performanz der verschiedenen Methoden verglichen werden.

Die klassische Methode. Als vergleichende klassische Methode wurde das Programm DataModel[6] der FH Osnabrück, das unter der Leitung von Prof. Gervens entwickelt wurde, verwendet (Abbildung 3.10). Es verwaltet zweistufige Feed-Forward-Netze mit sigmoiden Aktivierungsfunktionen, die mit Hilfe des Bärmann-Algorithmus [BaeBie] trainiert werden. Dieser arbeitet zwar iterativ, zeichnet sich aber empirisch durch ein sehr zuverlässiges Generalisierungsverhalten aus, d.h. er terminiert (in der Praxis) nicht in einem lokalen Suboptimum und benötigt auch kein Early-Stopping. Der einzige einstellbare Netzparameter ist die Anzahl der inneren Knoten, die zwar automatisch durch eine Lern/Testset-Einteilung bestimmt werden kann, wovon aber bei den Auswertungen in diesem Abschnitt kein Gebrauch gemacht wurde.

Die vietensche Methode. [7]Die Ergebnisse mit DataModel wurden zunächst mit den bayesschen Methoden nach MacKay (Kapitel 2) verglichen (Abbildungen 3.11 und 3.12). Dazu wurde die Implementierung von Frau Vieten [Vieten] verwendet. Sie verwaltet gleiche Netzstrukturen wie DataModel, also zweistufige Feed-Forward-Netze, fügt den Gewichten jedoch eine Regularisierung hinzu. Der Zusammenhang zwischen dem Gewichtsregularisierungsfaktor σ_{w}, der in den Grafiken auf der Abszisse dargestellt ist, und der Konstanten α, die in Abschnitt 2.2 die a priori Verteilung der Gewichte beschreibt, ist $\alpha = \sigma_{\mathrm{w}}^{-2}$. Der Messfehler s der Trainingsdaten wird in der vietenschen Implementierung global für alle Trainingsdaten angenommen und automatisch nach Bayes trainiert[8]. Der Zusammenhang zwischen ihm und der Konstanten β ist $\beta = s^{-2}$.

Betrachtet man beide Verfahren, kann eine gute Generalisierung einen Testsetfehler[9] im Bereich von $0,32 \ldots 0,34$ erreichen. Beide Verfahren leisten dies. Man beachte dabei, dass der Testsetfehler trotz der recht großen Anzahl an Datensätzen im Testset eine verrauschte Größe ist.

[6] `http://www.fh-osnabrueck.de/05_angew_forschung/01forschung/fb_eui/eui02.pdf`

[7] Der hier gewählte Name leitet sich von der Diplomandin Frau Vieten ab.

[8] Genau genommen besteht der angenommene Messfehler sowohl aus dem individuell vorgegebenen Messfehler s_n als auch dem globalen trainierten Messfehler s. Da jedoch die individuellen Messfehler ohnehin alle gleich sind, spielt diese Eigenschaft des Trainingsalgorithmus hier keine Rolle.

[9] Das quadratische Mittel der Differenzen zwischen Mess- und Prognosewert über allen Datensätzen des Testsets.

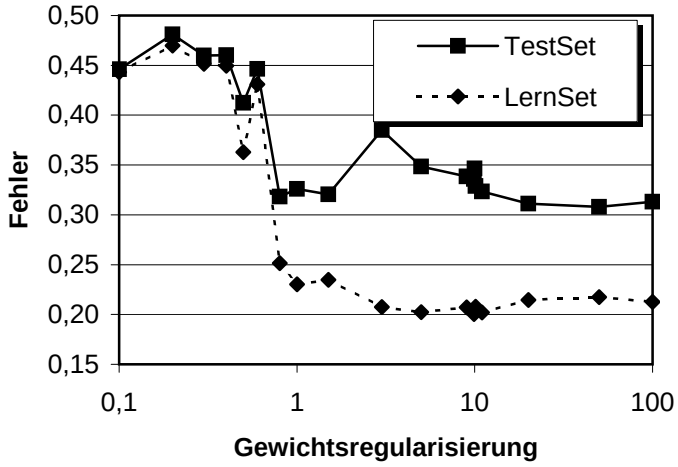

Abbildung 3.11: Mittlerer quadratischer Fehler bei [Vieten] mit globalem Fehler und 6 inneren Knoten

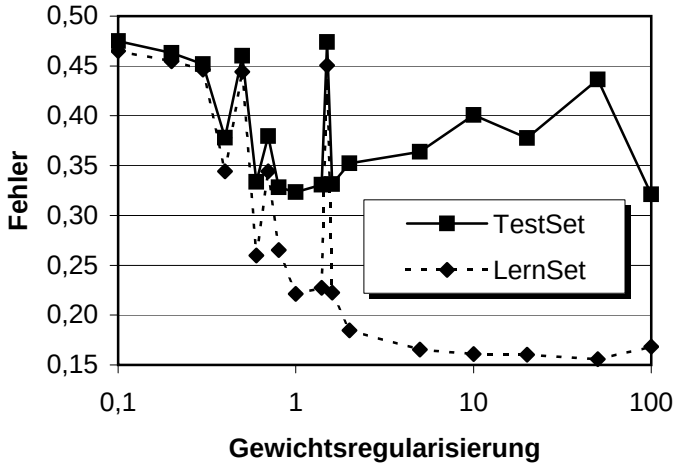

Abbildung 3.12: Mittlerer quadratischer Fehler bei [Vieten] mit globalem Fehler und 10 inneren Knoten

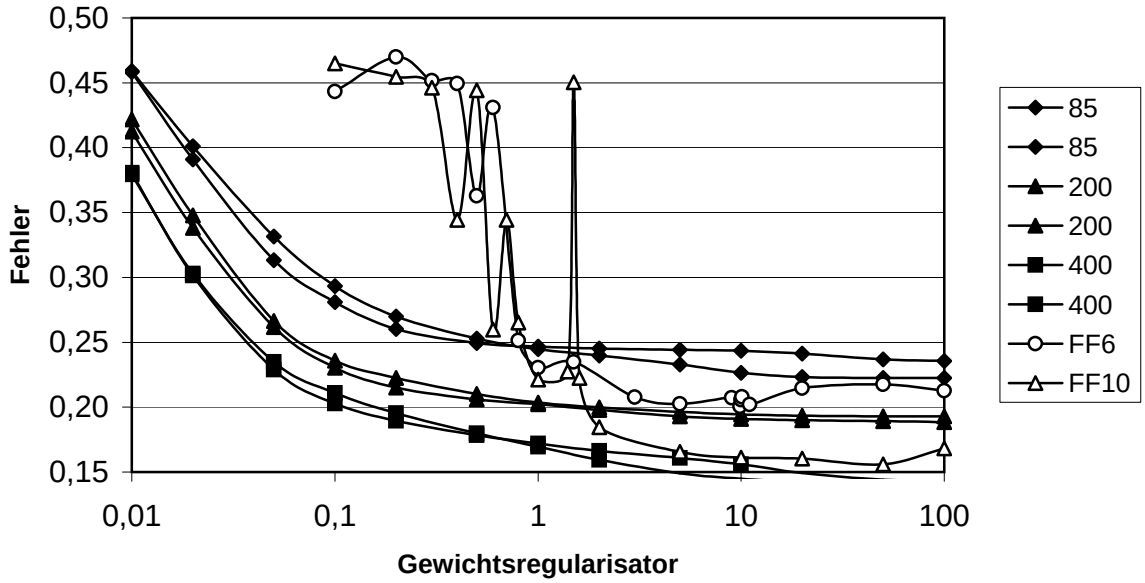

Abbildung 3.13: Fehler des Lernsets als Funktion von σ_w. Die Kurven 85, 200 und 400 bezeichnen GLNe mit entsprechend vielen Basisfunktionen, wobei je zwei Netze gleichen Typs, aber unterschiedlichen Zufallselementen erzeugt wurden. Die Kurven FF6 und FF10 entsprechen dem vietenschen Verfahren mit 6 bzw. 10 inneren Knoten. Die Gewichtsregularisierung auf der Abszisse ist zwischen den beiden Verfahren nicht vergleichbar.

Für beide Verfahren sind auf den Abszissen diejenigen Netzparameter dargestellt, die die Netzkomplexität bestimmen. Bei DataModel steigt die Komplexität mit der Anzahl der inneren Knoten, die die Anzahl der Gewichte bestimmt. Beim vietenschen Ansatz steigt die Komplexität mit der Gewichtsregularisierung σ_w.

Da bei DataModel der Testsetfehler bei 6 inneren Knoten minimal war, wurden auch 6 Knoten für den vietenschen Ansatz gewählt. Genau genommen sind 6 innere Knoten bei DataModel direkt äquivalent zu 6 inneren Knoten und $\sigma_w \to \infty$ bei [Vieten]. Wenn σ_w endlich gewählt wird, entspricht dies einer höheren Anzahl von Gewichten, weshalb in der zweiten Untersuchung 10 innere Knoten gewählt wurden.

Bei beiden Verfahren ist die Abhängigkeit des Lern- und Testsetfehlers von der Netzkomplexität wie erwartet zu beobachten. Zu einfache Netze führen zu einem hohen Lern- und Testsetfehler: das Netz ist nicht in der Lage, die Komplexität des zugrunde liegenden Phänomens ausreichend gut zu approximieren. Bei komplexen Netzen können beide Verfahren den Lernsetfehler wie zu erwarten immer weiter senken. DataModel verhält sich dabei stabil und senkt den Lernsetfehler nahezu monoton. [Vieten] dagegen konvergiert teilweise in lokalen Minima, was zu Ausreißern nach oben führt. Dabei ist der Testsetfehler auch immer mitbetroffen.

Das Verhalten des Testsetfehlers bei komplexeren Netzen ist bei allen Untersuchungen verschieden, jedoch immer so wie prinzipiell erwartet. Bei DataModel steigt der Testsetfehler in der Tendenz sehr leicht an, was auf einen leichten Overfitting-Effekt hindeutet. Verglichen mit anderen klassischen Methoden verhält sich der Bärmann-Algorithmus also auch dann noch gutmütig. Bei [Vieten] mit 6 inneren Knoten fällt der Testsetfehler tendenziell, wobei einige Ausreißer nach oben, wahrscheinlich suboptimale Minima, zu beklagen sind. Dies verwundert nicht, denn die Netzkomplexität ist durch nur 6 innere Knoten beschränkt, weshalb Overfitting nicht wirklich auftreten kann. Bei [Vieten] mit 10 inneren Knoten sieht dies anders aus: 10 Knoten sind offensichtlich schon zu viele und dem kann nur eine entsprechende Gewichtsregularisierung entgegenwirken. Daher steigt der Testsetfehler für große σ_w tendenziell an.

Generalisierte lineare Netze. Das dritte Verfahren besteht aus generalisierten linearen Netzen, kurz GLNen, nach Abschnitt 3.1. Auswertungen dazu sind in den Abbildungen 3.13 und 3.14 dargestellt.

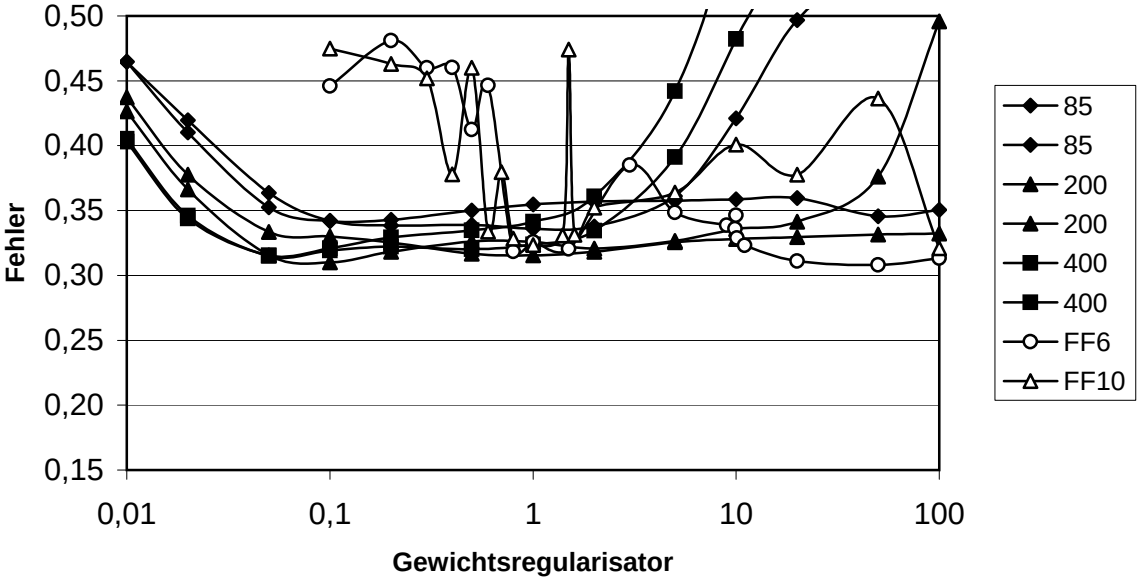

Abbildung 3.14: Fehler des Testsets als Funktion von σ_{w}. Die Bezeichnungen entsprechen denen aus Abbildung 3.13.

Als Basisfunktionen wurden ein Bias (konstant 1), zwölf lineare Basisfunktionen $(x_1, \ldots, x_{12})$ und weitere zufällig bestimmte sigmoide Basisfunktionen nach Gleichung 3.140, Seite 67, verwendet. Da die bei DataModel und [Vieten] verwendeten Feed-Forward-Netze bei 6 inneren Knoten genau 85 Gewichte besitzen, wurden auch bei den GLNen zunächst 85 Basisfunktionen gewählt. Um den Einfluss der Gewichtsregularisierung zu studieren, wurden daneben auch noch 200 und 400 Basisfunktionen getestet. Man beachte, dass durch die Gewichtsregularisierung auch das ansonsten unterbestimmte System (400 Gewichte bei 267 Trainingsdaten) praktikabel ist.

Die GLNe besitzen einen sowohl in der Anzahl der Basisfunktionen als auch in der Gewichtsregularisierung σ_{w} monoton sinkenden Lernsetfehler. Das ist auch nicht verwunderlich, denn sämtliche Berechnungen sind deterministisch und (im Rahmen der Numerik) exakt. Neben dem globalen Minimum w_{MP} der Fehlerfunktion gibt es keine weiteren lokalen Minima, außerdem ist w_{MP} stetig in σ_{w}. Bei gleicher Anzahl an Gewichten und vernachlässigbarer Gewichtsregularisierung können die Feed-Forward-Netze offenbar besser approximieren als die GLNe. Dies ist jedoch bereits theoretisch durch [Barron] untersucht worden und im Zusammenhang mit bayesschen Methoden nicht relevant.

Der Testsetfehler nimmt seinen minimalen Wert bei allen untersuchten GLN-Varianten bei einem ähnlich kleinen Wert wie bei den beiden anderen Methoden an. GLNe generalisieren demnach genauso gut. Wie nicht anders zu erwarten war, ist die testset-optimale Gewichtsregularisierung σ_{w} umso kleiner (die Gewichte sind a priori betragsmäßig kleiner), je größer die Anzahl der Gewichte ist. Die optimale Komplexität des Netzes reguliert sich also automatisch selbst.

Der Einfluss des Zufalls in den Konstanten der Basisfunktionen ist offensichtlich nicht groß. Dies ist ein wichtiges Ergebnis, denn ansonsten müsste der Einsatz von Komitees erwogen werden, deren Mitglieder sich in den Zufallselementen der Basisfunktionen unterscheiden, was eine deutliche Erhöhung diverser Rechenzeiten nach sich ziehen würde. Ein Ausnahme bilden lediglich sehr hohe Gewichtsregularisierungsfaktoren σ_{w} bei geringen Anzahlen von Basisfunktionen, was für die Suche nach dem optimalen Gewichtsregularisierungsfaktor aber irrelevant ist.

Der Einfluss der Anzahl der Basisfunktionen ist zwar größer als der des Zufalls, jedoch kann mit jeder Anzahl von Basisfunktionen, die groß genug ist (hier etwa 85), eine ausreichend gute Generalisierung erreicht werden. Auch dieses Ergebnis ist wichtig, denn die Anzahl der Basisfunktionen muss nicht in einem Komitee variiert werden, es genügt ein Netz mit ausreichend vielen Basisfunktionen.

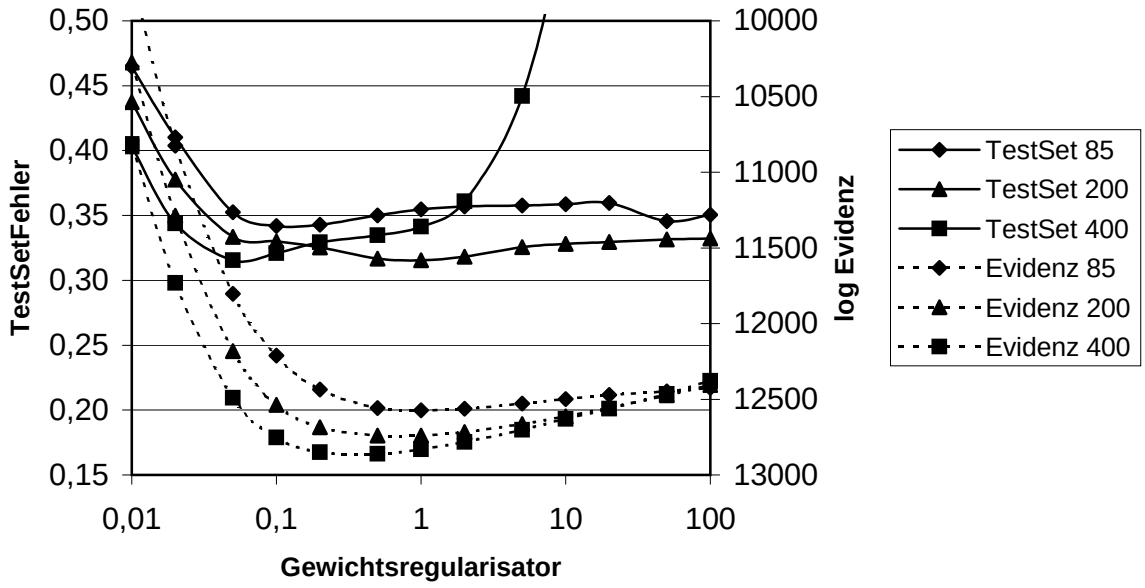

Abbildung 3.15: Fehler des Testsets und Logarithmus der Evidenz als Funktionen von σ_w bei drei verschiedenen GLNen.

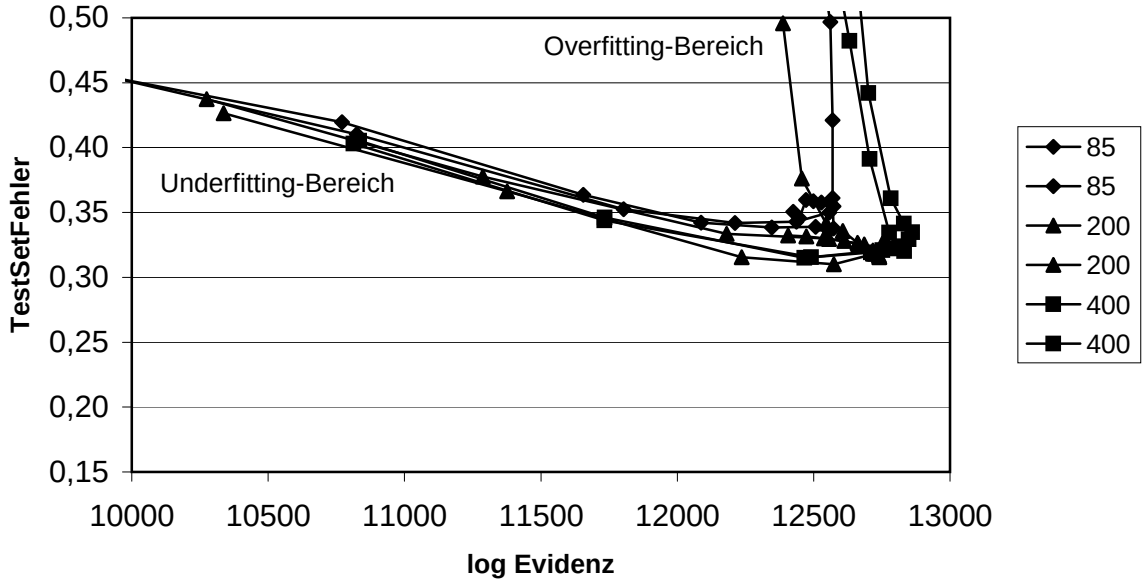

Abbildung 3.16: Korrelation zwischen der Evidenz und dem Testsetfehler für verschiedene GLNe.

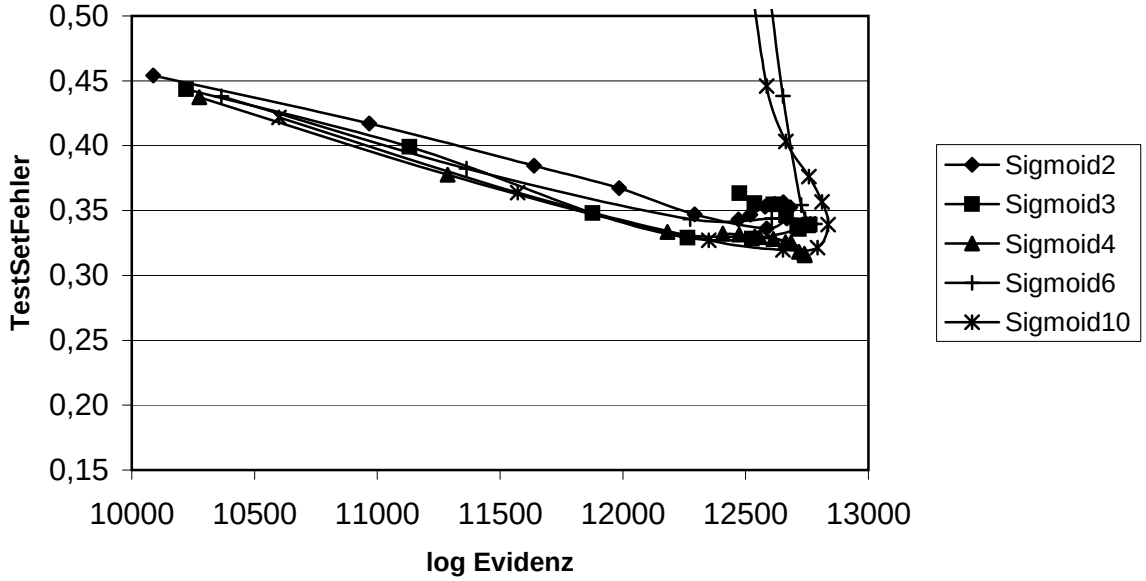

Abbildung 3.17: Korrelation zwischen der Evidenz und dem Testsetfehler für verschiedene sigmoide Basisfunktionen.

Die Evidenz. Die automatische Bestimmung der Gewichtsregularisierung nach Abschnitt 3.2 basiert auf der Evidenz $p(D|\sigma_{\mathrm{w}})$ (Gleichung 3.25). Die Evidenz kann für die GLNe nach Gleichung 3.27 berechnet werden. Für klassische Verfahren ist natürlich keine Evidenz definiert, die vietensche Implementierung bietet keine Berechnung an.

Die Verwendung der Evidenz soll ein Testset überflüssig machen. Dazu muss sie jedoch entsprechend mit dem Testsetfehler korreliert sein. Abbildung 3.15 zeigt den Testsetfehler und die Evidenz in Abhängigkeit der Gewichtsregularisierung für verschieden viele Basisfunktionen. Die berechneten Evidenzen sind zwischen den verschiedenen Netzen nicht vergleichbar, da entsprechende konstante Faktoren weggelassen wurden.

Man sieht, dass dort, wo die Evidenz maximal ist (Achtung: Skala der Log-Evidenz ist invers dargestellt), auch der jeweilige Testsetfehler gering ist. Der Zusammenhang ist allerdings nicht ganz perfekt, was aber am inhärenten Rauschen des Testsetfehlers liegen dürfte, der beispielsweise bei 85 Basisfunktionen zwei getrennte lokale Minima aufweist.

Abbildung 3.16 stellt die Evidenz und den Testsetfehler direkt in Verbindung dar. Man sieht die starke Korrelation der beiden Größen, die eher wenig von der Anzahl der Basisfunktionen abhängt. Insbesondere liegt das jeweilige Maximum der Evidenz immer in einem Bereich mit vergleichsweise geringem Testsetfehler.

Auffällig ist jedoch, dass die Evidenz und der Testsetfehler nicht linear miteinander korreliert sind, sondern sich zwei Arme der Kurve abzeichnen. Während bei zu starker Regularisierung (kleines σ_{w}) die Evidenz sehr schnell sehr klein wird, schnellt der Testsetfehler bei zu geringer Regularisierung in die Höhe. Für die praktische Anwendung (Abschnitt 3.2), insbesondere die automatische Bestimmung von σ_{w}, ist dieses Verhalten allerdings nicht relevant.

Verschiedene Basisfunktionen. Die Wahl der Basisfunktionen in Abschnitt 3.4.1 basiert auch auf dem empirischen Vergleich verschiedener Sätze von Basisfunktionen. Dazu wurden Netze mit einem Bias, 12 linearen Basisfunktionen und je 187 nicht-linearen Basisfunktionen verglichen, die Ergebnisse sind in den Abbildungen 3.17 und 3.18 dargestellt. Folgende nicht-lineare Basisfunktionen wurden verwendet:

$$g_{\mathrm{Sigmoid2}}(x) = \frac{1}{1+\exp\left(3\left(c+\sum_{i=1}^{2} d_i x_{j_i}\right)\right)} \tag{3.138}$$

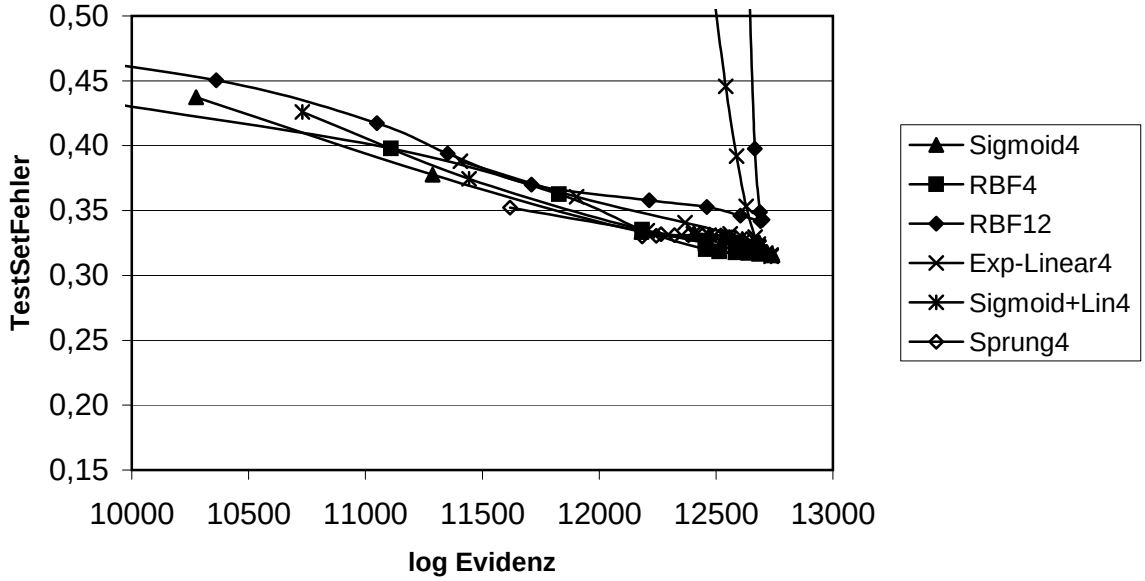

Abbildung 3.18: Korrelation zwischen der Evidenz und dem Testsetfehler für verschiedene Grundtypen von Basisfunktionen.

$$g_{\text{Sigmoid3}}(x) = \frac{1}{1+\exp\left(3\left(c+\sum_{i=1}^{3} d_i x_{j_i}\right)\right)} \tag{3.139}$$

$$g_{\text{Sigmoid4}}(x) = \frac{1}{1+\exp\left(3\left(c+\sum_{i=1}^{4} d_i x_{j_i}\right)\right)} \tag{3.140}$$

$$g_{\text{Sigmoid6}}(x) = \frac{1}{1+\exp\left(3\left(c+\sum_{i=1}^{6} d_i x_{j_i}\right)\right)} \tag{3.141}$$

$$g_{\text{Sigmoid10}}(x) = \frac{1}{1+\exp\left(3\left(c+\sum_{i=1}^{10} d_i x_{j_i}\right)\right)} \tag{3.142}$$

$$g_{\text{RBF4}}(x) = \exp\left(-\sum_{i=1}^{4}(x_{j_i}-d_i)^2\right) \tag{3.143}$$

$$g_{\text{RBF12}}(x) = \exp\left(-\frac{1}{6}\sum_{i=1}^{12}(x_i-d_i)^2\right) \tag{3.144}$$

$$h_4(x) := c+\sum_{i=1}^{4} d_i x_{j_i}$$

$$g_{\text{Exp-Linear4}}(x) = \begin{cases} \exp(h_4(x)), & \text{falls } h_4(x)<0 \\ 1+h_4(x), & \text{sonst} \end{cases} \tag{3.145}$$

$$g_{\text{Sigmoid+Lin4}}(x) = \frac{1}{1+\exp(3\cdot h_4(x))}+\frac{1}{2}h_4(x) \tag{3.146}$$

$$g_{\text{Sprung4}}(x) = \begin{cases} -1, & \text{falls } h_4(x)<0 \\ 1, & \text{sonst} \end{cases} \tag{3.147}$$

Dabei sind die Zufallszahlen $c, d_1, d_2, \ldots$ standardnormalverteilt und die Indizes $j_1, j_2, \ldots \in \{1, \ldots, 12\}$ gleichverteilt.

Alle getesteten Sätze von Basisfunktionen besitzen einen zufriedenstellend kleinen minimalen Testsetfehler, der sich zudem in der Nähe des Maximums der Evidenz befindet. Daher kann gesagt werden, dass alle diese Basisfunktionen ein gutes Generalisierungsverhalten des Netzes ermöglichen.

Dieses Ergebnis erstaunt insbesondere bei den Sprung4-Funktionen, die eine unstetige und gebietsweise konstante Netzfunktion implizieren. In einem praktischen System würden derartige Basisfunktionen kaum je eingesetzt. Eine mögliche Erklärung für das trotzdem gute Generalisierungsverhalten liegt aber in der Anzahl der Basisfunktionen: ist diese groß genug, kann die wahre Funktion gut genug approximiert werden.

Kapitel 4

Erweiterte Modelle

Dieses Kapitel beschreibt detailliert drei wichtige im Projekt verwendete Modelle, die die bisher vorgestellten bayesschen Methoden erweitern. Dabei werden die in Kapitel 3 beschriebenen Netze als Berechnungseinheiten zur Prognose diverser Größen verwendet.

In der Literatur wird der Begriff Modell im Wesentlichen synonym mit den Begriffen Netzfunktion oder Netzstruktur verwendet. Hier bezeichnet ein Modell jedoch einen Teilaspekt eines auf realen Daten basierenden Problems, dessen mathematische Formulierung und algorithmische Lösung.

4.1 Kooperation von Netzen

Die Kooperation von Netzen ist — im Sinne dieser Arbeit — eine Menge von unabhängigen Netzen, die sich zusammen ähnlich einem einzelnen Netz verhalten. Sie verteilt ihre Trainingsdaten disjunkt an ihre Netze und kombiniert die Prognosen der Netze wieder zu einer Gesamtprognose. Die Kooperation selbst besitzt kein „Wissen“ in irgendeiner Form und wird auch nicht trainiert.

4.1.1 Motivation

Die Vorteile der Kooperation mit ihren vielen kleinen Netzen gegenüber einem großen universalen Netz sind Folgende:

- Die Kooperation kann auch in sehr großen Eingangsräumen mit sehr vielen Trainingsdaten, die in Clustern angeordnet sind, Prognosen in geringer Rechenzeit berechnen. Jedes einzelne Netz kann dabei vergleichsweise einfach gestaltet sein, etwa eine nur geringe Anzahl von Basisfunktionen besitzen.
- Sie kann auf effiziente Weise verteilte Eingangsvariablen in den Trainingsdaten verarbeiten. Verteilte Eingangsvariablen („missing values“, „incomplete patterns“) sind unbekannte Komponenten des Vektors der Messstellen.
- Sie ermöglicht eine effiziente Unterhaltung eines Prognosesystems, das auf ständig aktualisierten und ergänzten Trainingsdaten arbeitet: ändert sich ein Trainingsdatensatz, so muss lediglich das zugehörige Netz neu trainiert werden.
- Die Einteilung der Trainingsdaten auf die Netze hilft dem Anwender später Bereiche für die Prognose zu finden, in denen belastbare Aussagen (relativ kleine Prognosefehler) gemacht werden können, wenn die Trainingsdaten extrem geclustert sind. Die Netze können nämlich so gestaltet werden, dass sie die Cluster nachbilden.

Kapitel 5 beschäftigt sich unter anderem mit der praktischen Anwendung der Kooperation.

Die Kooperation ähnelt verschiedenen Verfahren, die bereits intensiv in der Literatur diskutiert wurden. Dazu zählen zunächst die „mixtures of experts“ und ihre Erweiterung „hierarchical mixtures of experts“: [AvnInt], [FriFinWai], [JacJorNow], [JacTanPen], [PenJacTan], [JiaTan1], [JiaTan2], [Moerland],

[RaoMilRos], [WatMacRob], [XuJorHin] und [Xu]. Wie die Kooperation auch sollen die einzelnen „experts“ die wahre Funktion in bestimmten Regionen des Eingangsraums beschreiben, man erhofft sich durch diese Lokalisierung eine verbesserte Generalisierungsfähigkeit. Im Unterschied zur Kooperation werden aber sowohl die einzelnen „experts“ als auch der oder die „gating experts“, die die Regionen bestimmten, gleichzeitig trainiert.

Andere, der Kooperation ähnliche Verfahren beschäftigen sich mit der Kombination von Prognosen verschiedener „experts“ (die nicht notwendigerweise neuronale Netze sein müssen) zu einer Gesamtaussage: [AlkKit], [Breiman], [ImpSal], [Kuncheva], [PerCoo]. Dabei werden die einzelnen „experts“ auf allen oder zumindest überschneidenden Trainingsdaten trainiert, sodass sich stochastische Abhängigkeiten zwischen den „experts“ ergeben. Im bayesschen Kontext ähnelt dies Komitees.

Wie alle Netze auch soll die Kooperation Prognosewerte $\mu(x)$ und Prognosevarianzen $\sigma^2(x)$ berechnen. Da das Verhalten der Prognosen der in Kapitel 3 eingeführten Netze kompliziert ist, werden hier vereinfachende Annahmen über das Verhalten von Netzen getroffen. Sowohl die Netze als auch die Kooperation sollen die folgenden drei Annahmen erfüllen:

$$\textbf{[Annahme 1]} \qquad E\left[\mu(x)|f\right] = f(x) \tag{4.1}$$

$$\textbf{[Annahme 2]} \qquad E\left[\sigma^2(x)|f\right] = VAR\left[\mu(x)|f\right] \tag{4.2}$$

$$\textbf{[Annahme 3]} \qquad \sigma^2(x) \text{ hängt nicht von den Trainingswerten } t_1, \ldots, t_N \text{ ab.} \tag{4.3}$$

Die Notation der wahren Funktion f als Bedingung der Erwartungswerte bzw. der Varianz bedeutet, dass hier der Erwartungswert bzw. die Varianz über alle möglichen Folgen von Trainingswerten $[t_1, \ldots, t_N]$, verteilt nach Gleichung 3.1, gebildet werden soll. Annahme 1 bedeutet dann anschaulich, dass die Prognosen im Mittel über viele Messungen (an den Stellen $x_1, \ldots, x_N$ mit den Messfehlern $s_1, \ldots, s_N$) den wahren Wert ergeben sollen. Annahme 2 drückt aus, dass das Netz die Abweichung zwischen seinem Prognosewert und dem wahren Wert erwartungstreu durch seinen Prognosefehler einschätzt. Annahme 3 ist eher technischer Natur und nötig, um die nachfolgenden Berechnungen durchführen zu können.

Alle drei Annahmen lassen sich aus den Gleichungen 3.100 und 3.101 herleiten. Während dies bei Annahme 3 offensichtlich ist, folgt hier die kurze Herleitung für Annahme 1

$$\begin{aligned}
E[\mu(x)|f] &= E\left[\left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} \sum_{i:x_i\approx x} t_i s_i^{-2} | f\right] \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} \sum_{i:x_i\approx x} s_i^{-2} E[t_i|f] \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} \sum_{i:x_i\approx x} s_i^{-2} f \\
&= f,
\end{aligned} \tag{4.4}$$

die linke Seite von Annahme 2

$$\begin{aligned}
E[\sigma^2(x)|f] &= E\left[\left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} | f\right] \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1}
\end{aligned} \tag{4.5}$$

und die rechte Seite von Annahme 2

$$\begin{aligned}
VAR[\mu(x)|f] &= E\left[(\mu(x) - E[\mu(x)|f])^2 | f\right] \\
&= E\left[(\mu(x) - f)^2 | f\right]
\end{aligned}$$

$$
\begin{aligned}
&= E\left[\left(\left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} \sum_{i:x_i\approx x} t_i s_i^{-2} - f\right)^2 |f\right] \\
&= E\left[\left(\left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} \sum_{i:x_i\approx x} (t_i - f) s_i^{-2}\right)^2 |f\right] \\
&= E\left[\left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-2} \left(\sum_{i:x_i\approx x} (t_i - f) s_i^{-2}\right) \left(\sum_{j:x_j\approx x} (t_j - f) s_j^{-2}\right) |f\right] \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-2} \sum_{i:x_i\approx x} \sum_{j:x_j\approx x} E[(t_i - f)(t_j - f) s_i^{-2} s_j^{-2} | f] \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-2} \sum_{i:x_i\approx x} E[(t_i - f)^2 s_i^{-4} | f] \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-2} \sum_{i:x_i\approx x} s_i^2 s_i^{-4} \\
&= \left(\sum_{i:x_i\approx x} s_i^{-2}\right)^{-1} . \qquad (4.6)
\end{aligned}
$$

Beim Training teilt die Kooperation ihre Trainingsdaten $D = \{t_1, \ldots, t_N\}$ in einzelne disjunkte Mengen auf, $D = D_1 \uplus \ldots \uplus D_J$, wobei über die Aufteilung in diesem Abschnitt keine weitere Annahme getroffen wird. Jede dieser Mengen D_j enthält die Trainingsdaten für ein Netz, das dann Prognosen $\mu_j(x)$ und $\sigma_j^2(x)$ berechnet ($j = 1, \ldots, J$). Die Disjunktheit der Aufteilung führt zur stochastischen Unabhängigkeit der Prognosen dieser Netze. Die folgenden drei Abschnitte beschäftigen sich nun mit der Herleitung der Gleichungen, die die Einzelprognosen der Netze wieder zu einer Gesamtprognose kombinieren.

4.1.2 Herleitung über eine Linearkombination

Bei diesem Ansatz soll der Gesamtprognosewert $\mu(x)$ eine Linearkombination der Einzelprognosewerte $\mu_1(x)$, …, $\mu_J(x)$ sein,

$$
\mu(x) = \sum_{j=1}^{J} \alpha_j \mu_j(x). \qquad (4.7)
$$

Die Koeffizienten $\alpha_1, \ldots, \alpha_J$ sollen dabei nicht von den Prognosewerten $\mu_1(x), \ldots, \mu_J(x)$ abhängen. Aus Annahme 1 für die Netze ergibt sich nun

$$
\begin{aligned}
f(x) &= E[\mu(x)|f] \\
&= E\left[\sum_{j=1}^{J} \alpha_j \mu_j(x)\right] \\
&= \sum_{j=1}^{J} \alpha_j E[\mu_j(x)] \\
&= \sum_{j=1}^{J} \alpha_j f(x). \qquad (4.8)
\end{aligned}
$$

Es überrascht nicht, dass sich hier

$$\sum_{j=1}^{J} \alpha_j = 1 \tag{4.9}$$

als hinreichende Bedingung für Annahme 1 für die Kooperation ergibt.

Da die Trainingsdaten der Netze paarweise stochastisch unabhängig sind, sind ihre Prognosewerte $\mu_1(x), \ldots, \mu_J(x)$ ebenfalls stochastisch unabhängig. Daher gilt für die rechte Seite von Annahme 2

$$\begin{aligned} VAR[\mu(x)|f] &= E\left[(\mu(x) - f(x))^2 | f\right] \\ &= E\left[\left(\left(\sum_{j=1}^{J} \alpha_j \mu_j(x)\right) - f(x)\right)^2 | f\right] \\ &= E\left[\left(\sum_{j=1}^{J} \alpha_j (\mu_j(x) - f(x))\right)^2 | f\right] \\ &= E\left[\sum_{j=1}^{J} \alpha_j^2 (\mu_j(x) - f(x))^2 | f\right] \\ &= \sum_{j=1}^{J} \alpha_j^2 \sigma_j^2(x). \end{aligned} \tag{4.10}$$

Die Annahmen 2 und 3 sind erfüllt, wenn

$$\sigma^2(x) = \sum_{j=1}^{J} \alpha_j^2 \sigma_j^2(x) \tag{4.11}$$

gewählt wird.

Bis hierher sind die Koeffizienten $\alpha_1, \ldots, \alpha_J$ mit Ausnahme der Nebenbedingung aus Gleichung 4.9 noch völlig unbestimmt. Sie sollen nun so gewählt werden, dass die Gesamtprognosevarianz $\sigma^2(x)$ minimal wird. Mit Hilfe der entsprechenden Lagrangefunktion (z.B. in [Bishop], appendix C) können sie leicht bestimmt werden:

$$L(\alpha_1, \ldots, \alpha_J, \lambda) := \sum_{j=1}^{J} \alpha_j^2 \sigma_j^2(x) + \lambda \left(1 - \sum_{j=1}^{J} \alpha_j\right). \tag{4.12}$$

Die Gleichungen

$$\forall j = 1, \ldots, J: \quad \frac{\partial}{\partial \alpha_j} L(\alpha_1, \ldots, \alpha_J, \lambda) = 2\alpha_j \sigma_j^2(x) - \lambda = 0 \tag{4.13}$$

$$\frac{\partial}{\partial \lambda} L(\alpha_1, \ldots, \alpha_J, \lambda) = 1 - \sum_{j=1}^{J} \alpha_j = 0 \tag{4.14}$$

werden durch

$$\alpha_j = \frac{\sigma_j^{-2}(x)}{\sum_{i=1}^{J} \sigma_i^{-2}(x)} \qquad \text{für } j = 1, \ldots, J \text{ und} \tag{4.15}$$

$$\lambda = \frac{2}{\sum_{i=1}^{J} \sigma_i^{-2}(x)} \tag{4.16}$$

eindeutig erfüllt. Man beachte, dass hieraus insbesondere $0 < \alpha_j < 1$ für $j = 1, \ldots, J$ folgt. Wie zu erwarten war, handelt es sich also bei der Berechnung des Gesamtprognosewerts in Gleichung 4.7 um eine Konvexkombination der Einzelprognosewerte. Insgesamt stellen damit die Gleichungen

$$\mu(x) = \left(\sum_{j=1}^{J} \sigma_j^{-2}(x)\right)^{-1} \cdot \left(\sum_{j=1}^{J} \sigma_j^{-2}(x)\mu_j(x)\right) \tag{4.17}$$

$$\sigma^2(x) = \left(\sum_{j=1}^{J} \sigma_j^{-2}(x)\right)^{-1} \tag{4.18}$$

die Zusammenfassung der Einzelprognosen zur Gesamtprognose dar. Sie definieren daher das Verhalten der Kooperation bei der Prognose.

4.1.3 Herleitung über die Annahme einer Normalverteilung

Die Prognose eines Netzes mit bayesschen Methoden besteht aus der Verteilung der Netzausgangsvariablen $t(x)$, deren Kennzahlen Erwartungswert und Varianz durch die Annahmen 1 und 2 beschrieben werden. Nimmt man an, dass die Netzausgaben bekannten Verteilungen unterliegen, kann man sie zu einer Gesamtverteilung kombinieren. Aufgrund der stochastischen Unabhängigkeit der Netze gilt

$$\begin{aligned} p(t(x)|D) &= \frac{p(t(x))}{p(D)} p(D|t(x)) \\ &= \frac{p(t(x))}{p(D)} \prod_{j=1}^{J} p(D_j|t(x)) \\ &= \frac{p(t(x))}{p(D)} \prod_{j=1}^{J} \frac{p(D_j)}{p(t(x))} p(t(x)|D_j) \\ &= p(t(x))^{-J+1} \prod_{j=1}^{J} p(t(x)|D_j), \end{aligned} \tag{4.19}$$

wobei angenommen wurde, dass die a priori Verteilungen der Ausgaben, $p(t(x))$, für alle Netze und die Kooperation identisch sind. Man erhält also auch für die Kooperation eine Ausgangsvariable $t(x)$ mit bekannter Verteilung.

Betrachtet man Gleichung 4.19, so ist offenbar die Funktionsweise der Kooperation durch die Festlegung der a priori Verteilung und der Verteilung der Prognosen der Netze eindeutig gegeben. Unsere Ausgangsvariable $t(x)$ hat Positions- bzw. Lagecharakter (siehe dazu [Berger], „location parameter“): jeder Wert aus $\mathbb{R}$ kann vorkommen und man kann von keinem Wert sagen, dass er wahrscheinlicher ist als ein anderer. Als a priori Verteilung wird daher eine Gleichverteilung angenommen, $p(t(x)) = const$, und es folgt

$$p(t(x)|D) = const \cdot \prod_{j=1}^{J} p(t(x)|D_j), \tag{4.20}$$

wobei die Konstante eindeutig durch die Nebenbedingung $\int p(t(x)|D)d(t(x)) = 1$ bestimmt ist.

Die Ausgaben der Netze entsprechen physikalischen Messungen, werden also als Normalverteilungen angenommen. Die in der Implementierung verwendeten und in Abschnitt 3.1 vorgestellten Netze prognostizieren tatsächlich Normalverteilungen, wobei der Prognosewert $\mu_j(x)$ und die Prognosevarianz $\sigma_j^2(x)$ die beiden natürlichen Kennzahlen dieser Verteilung sind. Daher ist

$$p(t(x)|D) = const \cdot \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi\sigma_j^2(x)}} \exp\left(-\frac{(t(x) - \mu_j(x))^2}{2\sigma_j^2(x)}\right)$$

$$
\begin{aligned}
&= const \cdot \exp\left(-\frac{1}{2}\sum_{j=1}^{J}\frac{(t(x)-\mu_j(x))^2}{\sigma_j^2(x)}\right)\\
&= const \cdot \exp\left(-\frac{1}{2}\left(t^2(x)\sum_{j=1}^{J}\sigma_j^{-2}(x) - 2t(x)\sum_{j=1}^{J}\sigma_j^{-2}\mu_j(x)\right)\right)\\
&= const \cdot \exp\left(-\frac{1}{2}\sum_{j=1}^{J}\sigma_j^{-2}(x)\cdot\left(t(x) - \left(\sum_{j=1}^{J}(\sigma_j^{-2}(x)\right)^{-1}\sum_{j=1}^{J}\sigma_j^{-2}\mu_j(x)\right)^2\right). \quad (4.21)
\end{aligned}
$$

Somit ist $t(x)|D$ offensichtlich ebenfalls normalverteilt ist, und zwar nach

$$
\mathcal{N}\left(\left(\sum_{j=1}^{J}\sigma_j^{-2}(x)\right)^{-1}\cdot\sum_{j=1}^{J}\sigma_j^{-2}(x)\mu_j(x), \left(\sum_{j=1}^{J}\sigma_j^{-2}(x)\right)^{-1}\right). \quad (4.22)
$$

Die hier konstruierte Kooperation ist identisch mit der aus Abschnitt 4.1.2. Sie ist diejenige, die im weiteren Verlauf verwendet wird und auch implementiert wurde.

4.1.4 Herleitung über die Annahme einer Log-Normalverteilung

Nicht alle Ansätze zur Herleitung des Kooperationsverhaltens liefern die Gleichungen 4.17 und 4.18, in diesem Abschnitt soll daher ein Gegenbeispiel präsentiert werden.

Einige physikalische Größen haben beispielsweise Skalierungscharakter (siehe dazu [Berger], „scale parameter“); sie sind immer echt positiv, wobei jede Größenordnung (Zehnerpotenz) gleich wahrscheinlich ist. Daher wird für sie eine a priori Verteilung mit der Dichte $p(t(x)) = const/t(x)$ angenommen. Die Netze liefern für derartige Größen log-normalverteilte Prognosen. Die Log-Normalverteilung hat für die Parameter $\tilde{t}$ und $\tilde{s}$ die Dichte

$$
p(t(x)) = \frac{1}{\sqrt{2\pi}\tilde{s}t(x)}\exp\left(-\frac{(\ln t(x)-\tilde{t})^2}{2\tilde{s}^2}\right). \quad (4.23)
$$

Für den Erwartungswert und die Varianz gilt nach [Müller]

$$
\begin{aligned}
E[t(x)] &= \exp\left(\tilde{t}+\frac{\tilde{s}^2}{2}\right) && (4.24)\\
VAR[t(x)] &= \exp(2\tilde{t}+\tilde{s}^2)\left(\exp(\tilde{s}^2)-1\right). && (4.25)
\end{aligned}
$$

Daraus ergeben sich die Parameter zu

$$
\begin{aligned}
\tilde{t} &= \ln E[t(x)] - \frac{1}{2}\ln\left(\frac{VAR[t(x)]}{E[t(x)]^2}+1\right) && (4.26)\\
\tilde{s}^2 &= \ln\left(\frac{VAR[t(x)]}{E[t(x)]^2}+1\right). && (4.27)
\end{aligned}
$$

Gleichung 4.19 gilt unabhängig von der Wahl der a priori und der Einzelprognose-Verteilungen und konkretisiert sich nun zu

$$
\begin{aligned}
p(t(x)|D) &= \left(\frac{const}{t(x)}\right)^{-J+1}\cdot\prod_{j=1}^{J}\frac{1}{\sqrt{2\pi}\tilde{s}_j t(x)}\exp\left(-\frac{(\ln t(x)-\tilde{t}_j)^2}{2\tilde{s}_j^2}\right)\\
&= const\cdot\frac{1}{t(x)}\exp\left(-\frac{1}{2}\sum_{j=1}^{J}\frac{(\ln t(x)-\tilde{t}_j)^2}{\tilde{s}_j^2}\right)
\end{aligned}
$$

$$= const \cdot \frac{1}{t(x)} \exp\left(-\frac{1}{2}\left(\ln^2 t(x) \sum_{j=1}^{J} \tilde{s}_j^{-2} - 2\ln t(x) \sum_{j=1}^{J} \tilde{s}_j^{-2}\tilde{t}_j\right)\right)$$

$$= const \cdot \frac{1}{t(x)} \exp\left(-\frac{1}{2}\sum_{j=1}^{J} \tilde{s}_j^{-2}\left(\ln t(x) - \left(\sum_{j=1}^{J} \tilde{s}_j\right)^{-1} \sum_{j=1}^{J} \tilde{s}_j^{-2}\tilde{t}_j\right)^2\right). \quad (4.28)$$

Die Prognose der Kooperation hat also wiederum log-normalverteilte Form. Ihre Parameter sind

$$\tilde{t} = \left(\sum_{j=1}^{J} \tilde{s}_j^{-2}\right)^{-1} \sum_{j=1}^{J} \tilde{s}_j^{-2}\tilde{t}_j \quad (4.29)$$

$$\tilde{s}^2 = \left(\sum_{j=1}^{J} \tilde{s}_j^{-2}\right)^{-1}. \quad (4.30)$$

Hiermit ist die Kooperation für Skalierungsgrößen bereits vollständig beschrieben. Sie berechnet zunächst aus den Parametern der Einzelnetzausgaben $\mu_j(x) = E[t(x)|D_j]$ und $\sigma_j^2(x) = VAR[t(x)|D_j]$ die Parameter der Log-Normalverteilungen $\tilde{t}_j$ und $\tilde{s}_j^2$ gemäß den Gleichungen 4.26 und 4.27 für $j = 1, \ldots, J$. Anschließend werden diese Parameter mit Hilfe der Gleichungen 4.29 und 4.30 kombiniert und die erhaltenen Werte über die Gleichungen 4.24 und 4.25 wieder in die üblichen Parameter der Gesamtprognose zurückverwandelt. Kombiniert man diese Gleichungen durch Einsetzen der entsprechenden Größen, um direkt von den Prognosen der Netze auf die Prognosen der Kooperation zu schließen, ergeben sich kaum Vereinfachungen. Daraus ist einfach zu schließen, dass die Parameter $E[t(x)|D]$ und $VAR[t(x)|D]$ keine natürliche und algorithmisch günstige Darstellung der Verteilung von Skalengrößen darstellen.

Wir haben gesehen, dass bei der Konstruktion der Kooperation sowohl eine a priori Verteilung als auch eine Verteilung der Prognosen der Netze festgelegt werden müssen. Diese beiden Festlegungen müssen aber zueinander passen und sollten dem Charakter der Ausgangsvariablen entsprechen.

Natürlich muss die Kooperation nicht nur eine Verteilung der Netzausgaben annehmen, vielmehr müssen die Netze diese auch intern realisieren. Es muss also das gesamte System von Netzen an einheitlichen Verteilungen festhalten. Viele Größen kann man aber durch eine bestimmte Transformation in eine Größe mit Positions- bzw. Lagecharakter überführen. Hat man etwa eine Skalierungsgröße $t(x)$ mit den Verteilungen

$$p(t(x)) = \frac{const}{t(x)} \quad (4.31)$$

$$p(t(x)|D) = \frac{1}{\sqrt{2\pi}\tilde{s}t(x)} \exp\left(-\frac{(\ln t(x) - \tilde{t})^2}{2\tilde{s}^2}\right), \quad (4.32)$$

so gilt für die transformierte Größe $\tau(x) = \ln t(x)$:

$$p(\tau(x)) = \frac{p(\ln t(x))}{\frac{\partial}{\partial t(x)} \ln t(x)} = \frac{const/t(x)}{1/t(x)} = const \quad (4.33)$$

$$p(\tau(x)|D) = \frac{p(\ln(t(x))}{\frac{\partial}{\partial t(x)} \ln t(x)} = \frac{\frac{1}{\sqrt{2\pi}\tilde{s}t(x)} \exp\left(-\frac{(\ln t(x) - \tilde{t})^2}{2\tilde{s}^2}\right)}{1/t(x)} = \frac{1}{\sqrt{2\pi}\tilde{s}} \exp\left(-\frac{(\tau(x) - \tilde{t})^2}{2\tilde{s}^2}\right). \quad (4.34)$$

Somit unterliegt $\tau(x)$ den Verteilungen einer Positionsgröße und es können einfachere Gleichungen benutzt werden. Dies vereinfacht wiederum die Implementierung.

In Abschnitt 5.4 werden Transformationen der Ausgangsvariablen zur Verbesserung der Generalisierungsfähigkeit diskutiert. Das soeben dargestellte Beispiel legt nun nahe, die Kooperation auf der Ebene der transformierten und nicht auf der Ebene der originalen Werte stattfinden zu lassen, wenn die Gleichungen 4.17 und 4.18 verwendet werden. Die Abtragungsgeschwindigkeit, hier $t(x)$, als wichtige Messgröße der

Korrosion schwankt beispielsweise um mehrere Zehnerpotenzen und hat somit überwiegend Skalierungscharakter. Es ist folglich besser, die Kooperation auf der transformierten[1] Abtragungsgeschwindigkeit, hier $\tau(x)$, durchzuführen.

4.2 Vergleich zwischen kooperierenden Netzen und einem Gesamtnetz

Nachdem im vorigen Abschnitt die Kooperation von Netzen eingeführt wurde und auch auf einige Vorteile verwiesen wurde, stellt sich nun die Frage nach möglichen Nachteilen der Kooperation. Dazu wird die Kooperation zweier Netze namens A und B mit einem Gesamtnetz namens AB verglichen, das auf der Vereinigungsmenge der Trainingsdaten von A und B trainiert wurde. Als Maßstab des Vergleichs dient dabei der Prognosefehler, da angenommen wird, dass er erwartungstreu die Abweichung zwischen dem Prognosewert und dem wahrem Wert schätzt.

Seien im Folgenden

$$D_A = \{(t_{A,n}, s_{A,n}, x_{A,n})_{n=1,\ldots,N_A}\} \qquad \text{und} \qquad D_B = \{(t_{B,n}, s_{B,n}, x_{B,n})_{n=1,\ldots,N_B}\} \tag{4.35}$$

zwei disjunkte Mengen von Trainingsdaten für die Netze A bzw. B; die Trainingsdaten von Netz AB sind dann $D_A \uplus D_B$. Alle Netze sind hier konkrete Netze nach Abschnitt 3.1 und besitzen identische Basisfunktionen $g(x)$.

4.2.1 Abschätzung der Prognosevarianzen

Die Prognosevarianz $\sigma^2_{AB}(x)$ des Gesamtnetzes AB soll mit der Prognosevarianz der Kooperation der Netze A und B

$$\sigma^2_K(x) = \frac{1}{\sigma_A^{-2}(x) + \sigma_B^{-2}(x)} \tag{4.36}$$

nach Gleichung 4.18 verglichen werden. Um die nötigen Berechnungen durchführen zu können, wird zunächst die Gewichtsregularisierung weggelassen, d.h. die a priori Gewichtsverteilung aller drei Netze wird als gleichverteilt angenommen ($p(w) = const$), was dem Grenzübergang $\sigma_\mathrm{w} \to \infty$ entspricht. Die entsprechenden Hesse-Matrizen bestehen dann nur noch aus dem datenabhängigen Teil,

$$A_A = \sum_{n=1}^{N_A} \frac{1}{s^2_{A,n}} g(x_{A,n}) g(x_{A,n})^T \tag{4.37}$$

$$A_B = \sum_{n=1}^{N_B} \frac{1}{s^2_{B,n}} g(x_{B,n}) g(x_{B,n})^T, \tag{4.38}$$

und für die Prognosevarianzen gilt

$$\sigma^2_A(x) = g(x)^T A_A^{-1} g(x) \tag{4.39}$$

$$\sigma^2_B(x) = g(x)^T A_B^{-1} g(x) \tag{4.40}$$

$$\sigma^2_{AB}(x) = g(x)^T (A_A + A_B)^{-1} g(x). \tag{4.41}$$

Dabei wird angenommen, dass die Matrizen A_A und A_B invertierbar sind, da sich ansonsten keine Prognosen berechnen ließen.

Zur Bestimmung der Relation zwischen den Prognosevarianzen wird nun die folgende Differenz berechnet:

$$\begin{aligned} \delta &:= \sigma^{-2}_{AB}(x) - \sigma^{-2}_K(x) \\ &= \frac{1}{g(x)^T (A_A + A_B)^{-1} g(x)} - \left(\frac{1}{g(x)^T A_A^{-1} g(x)} + \frac{1}{g(x)^T A_B^{-1} g(x)} \right). \end{aligned} \tag{4.42}$$

[1] Die implementierte Transformation der Abtragungsgeschwindigkeit ist komplizierter.

Da die Matrix A_B symmetrisch und positiv definit ist, existiert eine reguläre Matrix $B \in \mathbb{R}^{M\times M}$ mit der Eigenschaft $BB^T = A_B$ (siehe etwa in [PreTeuVet]). Es folgt

$$\begin{aligned}\delta &= \frac{1}{g(x)^T(B^T)^{-1}B^T(A_A+A_B)^{-1}BB^{-1}g(x)} - \frac{1}{g(x)^T(B^T)^{-1}B^TA_A^{-1}BB^{-1}g(x)} \\ &\quad - \frac{1}{g(x)^T(B^T)^{-1}B^TA_B^{-1}BB^{-1}g(x)} \\ &= \frac{1}{g(x)^T(B^T)^{-1}(B^{-1}A_A(B^T)^{-1}+I)^{-1}B^{-1}g(x)} - \frac{1}{g(x)^T(B^T)^{-1}B^TA_A^{-1}BB^{-1}g(x)} \\ &\quad - \frac{1}{g(x)^T(B^T)^{-1}B^{-1}g(x)}. \end{aligned} \tag{4.43}$$

Verwendet man die Abkürzungen $C := B^{-1}A_A(B^T)^{-1}$ und $v := B^{-1}g(x)$ erhält man

$$\delta = \frac{1}{v^T(C+I)^{-1}v} - \frac{1}{v^TC^{-1}v} - \frac{1}{v^Tv}. \tag{4.44}$$

Die Matrix C ist positiv definit, denn für alle Vektoren $\tilde{v} \in \mathbb{R}^M \setminus \{0\}$ gilt

$$\begin{aligned}\tilde{v}^TC\tilde{v} &= \tilde{v}^TB^{-1}A_A(B^T)^{-1}\tilde{v} \\ &= \left((B^T)^{-1}\tilde{v}\right)^T A_A \left((B^T)^{-1}\tilde{v}\right) > 0, \end{aligned} \tag{4.45}$$

da A_A positiv definit und B regulär ist. Daher existiert ein System $u_1, \ldots, u_M$ von orthonormalen Eigenvektoren mit zugehörigen positiven Eigenwerten $\lambda_1, \ldots, \lambda_M$ von C. Diese Eigenvektoren bilden eine Basis des $\mathbb{R}^M$ und der Vektor v kann daher als Linearkombination von ihnen

$$v = \sum_{i=1}^{M} \alpha_i u_i \tag{4.46}$$

mit Koeffizienten $\alpha_1, \ldots, \alpha_M \in \mathbb{R}$ dargestellt werden. Es folgt durch Einsetzen in Gleichung 4.44:

$$\begin{aligned}\delta &= \frac{1}{\left(\sum_{i=1}^{M}\alpha_iu_i^T\right)(C+I)^{-1}\sum_{i=1}^{M}\alpha_iu_i} - \frac{1}{\left(\sum_{i=1}^{M}\alpha_iu_i^T\right)C^{-1}\sum_{i=1}^{M}\alpha_iu_i} - \frac{1}{\left(\sum_{i=1}^{M}\alpha_iu_i^T\right)\sum_{i=1}^{M}\alpha_iu_i} \\ &= \frac{1}{\left(\sum_{i=1}^{M}\alpha_iu_i^T\right)\sum_{i=1}^{M}\alpha_i(C+I)^{-1}u_i} - \frac{1}{\left(\sum_{i=1}^{M}\alpha_iu_i^T\right)\sum_{i=1}^{M}\alpha_iC^{-1}u_i} - \frac{1}{\left(\sum_{i=1}^{M}\alpha_iu_i^T\right)\sum_{i=1}^{M}\alpha_iu_i}. \end{aligned} \tag{4.47}$$

Hier werden nun die Eigenwerte und -vektoren der Matrizen $(C+I)^{-1}$ und C^{-1} benötigt, die wie folgt berechnet werden können: sei λ Eigenwert zum Eigenvektor u von C, dann ist

$$\begin{aligned}Cu &= \lambda u \\ \lambda^{-1}C^{-1}Cu &= \lambda^{-1}C^{-1}\lambda u \\ \lambda^{-1}u &= C^{-1}u \end{aligned} \tag{4.48}$$

und

$$\begin{aligned}Cu &= \lambda u \\ Cu + Iu &= \lambda u + u \\ (C+I)u &= (\lambda+1)u \\ (\lambda+1)^{-1}u &= (C+I)^{-1}u. \end{aligned} \tag{4.49}$$

Setzt man diese Ergebnisse nun in Gleichung 4.47 ein, erhält man

$$\delta = \frac{1}{\left(\sum_{i=1}^{M}\alpha_i u_i^T\right)\sum_{i=1}^{M}\alpha_i\frac{1}{\lambda_i+1}u_i} - \frac{1}{\left(\sum_{i=1}^{M}\alpha_i u_i^T\right)\sum_{i=1}^{M}\alpha_i\frac{1}{\lambda_i}u_i} - \frac{1}{\left(\sum_{i=1}^{M}\alpha_i u_i^T\right)\sum_{i=1}^{M}\alpha_i u_i}. \quad (4.50)$$

Aufgrund der Orthonormalität der Vektoren $u_1, \ldots, u_M$ lässt sich diese Gleichung weiter vereinfachen:

$$\delta = \frac{1}{\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i+1}} - \frac{1}{\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i}} - \frac{1}{\sum_{i=1}^{M}\alpha_i^2}. \quad (4.51)$$

Um das Vorzeichen von δ zu ermitteln, wird zur weiteren Betrachtung δ mit der Konstanten

$$c_1 := \left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i+1}\right)\left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_1}\right)\left(\sum_{i=1}^{M}\alpha_i^2\right) > 0 \quad (4.52)$$

multipliziert:

$$\begin{aligned}
c_1\delta &= \left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i}\right)\left(\sum_{i=1}^{M}\alpha_i^2\right) - \left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i+1}\right)\left(\sum_{i=1}^{M}\alpha_i^2\right) - \left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i+1}\right)\left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i}\right) \\
&= \frac{1}{2}\Bigg(\left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i}\right)\left(\sum_{j=1}^{M}\alpha_j^2\right) + \left(\sum_{j=1}^{M}\frac{\alpha_j^2}{\lambda_j}\right)\left(\sum_{i=1}^{M}\alpha_i^2\right) - \left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i+1}\right)\left(\sum_{j=1}^{M}\alpha_j^2\right) \\
&\qquad - \left(\sum_{j=1}^{M}\frac{\alpha_j^2}{\lambda_j+1}\right)\left(\sum_{i=1}^{M}\alpha_i^2\right) - \left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i+1}\right)\left(\sum_{j=1}^{M}\frac{\alpha_j^2}{\lambda_j}\right) - \left(\sum_{j=1}^{M}\frac{\alpha_j^2}{\lambda_j+1}\right)\left(\sum_{i=1}^{M}\frac{\alpha_i^2}{\lambda_i}\right)\Bigg) \\
&= \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i^2\alpha_j^2\left(\underbrace{\frac{1}{\lambda_i}+\frac{1}{\lambda_j}-\frac{1}{\lambda_i+1}-\frac{1}{\lambda_j+1}-\frac{1}{(\lambda_i+1)\lambda_j}-\frac{1}{(\lambda_j+1)\lambda_i}}_{\gamma(i,j)}\right). \quad (4.53)
\end{aligned}$$

Das Vorzeichen dieses Ausdrucks hängt von den Vorzeichen der Terme $\gamma(i,j)$ für $i,j = 1,\ldots,M$ ab. Um diese zu untersuchen wird $\gamma(i,j)$ mit der Konstanten $c_2(i,j) := \lambda_i(\lambda_i+1)\lambda_j(\lambda_j+1) > 0$ multipliziert:

$$\begin{aligned}
c_2(i,j)\gamma(i,j) &= (\lambda_i+1)\lambda_j(\lambda_j+1) + \lambda_i(\lambda_i+1)(\lambda_j+1) \\
&\qquad -\lambda_i\lambda_j(\lambda_j+1) - \lambda_i(\lambda_i+1)\lambda_j - \lambda_i(\lambda_j+1) - (\lambda_i+1)\lambda_j \\
&= \lambda_i\lambda_j^2 + \lambda_i\lambda_j + \lambda_j^2 + \lambda_j + \lambda_i^2\lambda_j + \lambda_i^2 + \lambda_i\lambda_j + \lambda_i \\
&\qquad -\lambda_i\lambda_j^2 - \lambda_i\lambda_j - \lambda_i^2\lambda_j - \lambda_i\lambda_j - \lambda_i\lambda_j - \lambda_i - \lambda_i\lambda_j - \lambda_j \\
&= \lambda_j^2 + \lambda_i^2 - 2\lambda_i\lambda_j \\
&= (\lambda_i - \lambda_j)^2. \quad (4.54)
\end{aligned}$$

Damit ist das Vorzeichen von δ geklärt, denn es gilt folgende Argumentationskette: $c_2(i,j)\gamma(i,j) \geq 0$ für alle $i,j = 1,\ldots,M$, daher $\gamma(i,j) \geq 0$ für alle diese i,j, daher $c_1\delta \geq 0$, daher $\delta \geq 0$ und damit schließlich $\sigma_{AB}^2(x) \leq \sigma_K^2(x)$.

Es wurde gezeigt, dass sich das Gesamtnetz immer mindestens so sicher ist wie die Kooperation. Dies ist kein verwunderliches Resultat, denn die Generalisierungsfähigkeit eines Netzes hängt sehr von der Zusammenstellung der Trainingsdaten ab. Wer im Rhein Fische fangen möchte würde dazu wohl auch nicht die Vorschläge eines Forellenzüchters A (Teiche anlegen) und eines Hochseefischers B (Hochseekutter verwenden) kombinieren, sondern einen allgemeinen Fischereifachmann AB befragen.

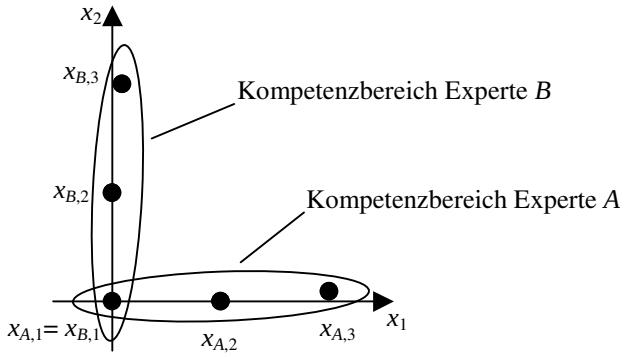

Abbildung 4.1: Beispiel zur Kooperation von Netzen

4.2.2 Gleichheit der Prognosevarianzen

Nachdem gezeigt wurde, dass die Kooperation nie einen Vorteil im Prognosefehler gegenüber dem Gesamtnetz bringt, stellt sich natürlich sofort die Frage, wann zumindest Gleichheit gilt. Nach Gleichung 4.54 ist $\gamma(i,j) \geq 0$ für alle $i,j = 1,\ldots,M$. Daher verschwindet δ gemäß Gleichung 4.53 genau dann, wenn alle Summanden, bestehend aus $\gamma(i,j)$ und den Vorfaktoren α_i^2 und α_j^2, verschwinden. Es gelten folgende äquivalente Aussagen:

$$\begin{aligned}
\delta = 0 \quad &\iff \quad \forall i,j = 1,\ldots,M \ : \quad \alpha_i = 0 \vee \alpha_j = 0 \vee \gamma(i,j) = 0 \\
&\iff \quad \forall i,j = 1,\ldots,M \ : \quad \alpha_i = 0 \vee \alpha_j = 0 \vee \lambda_i = \lambda_j \\
&\iff \quad \exists \lambda \in \mathbb{R} \ : \quad \forall i = 1,\ldots,M \ : \quad \alpha_i = 0 \vee \lambda_i = \lambda \\
&\iff \quad \exists \lambda \in \mathbb{R} \ : \quad v \in \mathrm{Span}\{u_i | \lambda_i = \lambda\} \\
&\iff \quad \exists \lambda \in \mathbb{R} \ : \quad v \text{ ist Eigenvektor von } C \text{ zum Eigenwert } \lambda \\
&\iff \quad \exists \lambda \in \mathbb{R} \ : \quad Cv = \lambda v \\
&\iff \quad \exists \lambda \in \mathbb{R} \ : \quad B^{-1}A_A(B^T)^{-1}B^{-1}g(x) = \lambda B^{-1}g(x) \\
&\iff \quad \exists \lambda \in \mathbb{R} \ : \quad A_A A_B^{-1} g(x) = \lambda g(x). \qquad (4.55)
\end{aligned}$$

Die Prognosevarianzen sind also genau dann gleich, wenn $g(x)$ ein Eigenvektor von $A_A A_B^{-1}$ ist.

Um dies praktisch zu interpretieren betrachten wir das folgende einfache Beispiel einer affin linearen Regression im $\mathbb{R}^2$, siehe dazu Abbildung 4.1. Die Basisfunktionen bestehen aus einem Bias und einer linearen Komponente in jeder Richtung:

$$g\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) \quad = \quad \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}. \qquad (4.56)$$

Drei Messstellen

$$x_{A,1} \ = \ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad x_{A,2} \ = \ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad x_{A,3} \ = \ \begin{pmatrix} 2 \\ 0,1 \end{pmatrix} \qquad (4.57)$$

bilden die Trainingsstellen des Netzes A und drei weitere Messstellen

$$x_{B,1} \ = \ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad x_{B,2} \ = \ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad x_{B,3} \ = \ \begin{pmatrix} 0,1 \\ 2 \end{pmatrix} \qquad (4.58)$$

die Trainingsstellen des Netzes B. Die Trainingsstellen liegen nicht alle auf den Koordinatenachsen, damit die von ihnen aufgespannten Räume der Netze nicht singulär und daher die resultierenden Matrizen invertierbar sind. Die Messfehler sollen der Einfachheit halber alle gleich groß sein.

$$s_{A,1} \ = \ s_{A,2} \ = \ s_{A,3} \ = \ s_{B,1} \ = \ s_{B,2} \ = \ s_{B,3} \ = \ 1 \qquad (4.59)$$

x	$\sigma_A(x)$	$\sigma_B(x)$	$\sigma_{AB}(x)$	$Q(x)$	
$\tilde{x}_1$	0,923	0,923	0,653	1,0	Anfrage in beiden Kompetenzbereichen
$(0\ \ 0)^T$	1,0	1,0	0,639	1,106	
$\tilde{x}_2$	0,740	40,867	0,740	1,0	Netz B kann keine sichere Aussage machen
$(1\ \ 0)^T$	1,0	24,920	0,525	1,904	
$(2\ \ 0)^T$	2,236	49,406	0,901	2,479	
$(5\ \ 0)^T$	6,403	122,886	2,538	2,519	Anfrage im Extrapolationsbereich von A
$(10\ \ 1)^T$	14,177	244,133	5,609	2,523	
$(1\ \ 1)^T$	23,685	23,685	0,615	27,226	Anfrage liegt abseits der Kompetenzbereiche von A noch in B, aber im von beiden gemeinsam aufgespannten Bereich
$(2\ \ 2)^T$	46,957	46,957	1,470	22,588	
$(2\ \ 1)^T$	22,472	48,177	1,073	18,979	

Tabelle 4.1: Vergleich zwischen Kooperation und Gesamtnetz an einigen Beispielanfragestellen

Die resultierenden Matrizen ergeben sich nun zu

$$A_A = \begin{pmatrix} 3,00 & 3,00 & 0,10 \\ 3,00 & 5,00 & 0,20 \\ 0,10 & 0,20 & 0,01 \end{pmatrix} \quad \text{und} \quad A_A^{-1} = \begin{pmatrix} 1 & -1 & 10 \\ -1 & 2 & -30 \\ 10 & -30 & 600 \end{pmatrix} \tag{4.60}$$

$$A_B = \begin{pmatrix} 3,00 & 0,10 & 3,00 \\ 0,10 & 0,01 & 0,20 \\ 3,00 & 0,20 & 5,00 \end{pmatrix} \quad \text{und} \quad A_B^{-1} = \begin{pmatrix} 1 & 10 & -1 \\ 10 & 600 & -30 \\ -1 & -30 & 2 \end{pmatrix}. \tag{4.61}$$

Die weiteren Ergebnisse werden hier gerundet dargestellt. Die Matrix $A_A A_B^{-1}$ hat die Eigenvektoren

$$u_1 = \begin{pmatrix} 1,00 \\ -0,02 \\ -0,02 \end{pmatrix}, \qquad u_2 = \begin{pmatrix} 1,00 \\ 1,65 \\ 0,07 \end{pmatrix}, \qquad u_3 = \begin{pmatrix} 1,00 \\ 0,07 \\ 1,65 \end{pmatrix}, \tag{4.62}$$

die innerhalb ihrer Eigenräume so gewählt wurden, dass sie von den Basisfunktionen $g(x)$ erzeugt werden können. Sie entsprechen daher den Punkten

$$\tilde{x}_1 = \begin{pmatrix} -0,02 \\ -0,02 \end{pmatrix}, \qquad \tilde{x}_2 = \begin{pmatrix} 1,65 \\ 0,07 \end{pmatrix}, \qquad \tilde{x}_3 = \begin{pmatrix} 0,07 \\ 1,65 \end{pmatrix}. \tag{4.63}$$

Punkt $\tilde{x}_1$ liegt im Kompetenzbereich beider Netze A und B, während $\tilde{x}_2$ und $\tilde{x}_3$ jeweils im Kompetenzbereich eines Netzes und weit weg vom Kompetenzbereich des anderen Netzes liegen.

Tabelle 4.1 listet einige Anfragepunkte x auf und untersucht die dortigen Prognosefehler. Die Spalte $Q(x)$ bezeichnet den Quotient zwischen dem Prognosefehler der Kooperation und dem Gesamtnetz

$$Q(x) \quad := \quad \frac{\sigma_K(x)}{\sigma_{AB}(x)} \quad = \quad \frac{\left(\sigma_A^{-2}(x) + \sigma_B^{-2}(x)\right)^{-1/2}}{\sigma_{AB}(x)}. \tag{4.64}$$

An den Stellen $\tilde{x}_1$ und $\tilde{x}_2$ ist natürlich $Q(x) = 1$, an den übrigen gilt $Q(x) > 1$.

Man sieht, dass sich der Prognosefehler durch die Aufteilung der Trainingsdaten und das Verwenden der Kooperation nicht dramatisch erhöht, wenn Anfragen in der Nähe der Trainingsdaten oder deren Extrapolationsbereich gestellt werden. Dies dürfte in der Praxis der häufigste Fall sein. Nur wenn Prognosen abseits der Trainingsdaten gefordert werden, ist das Gesamtnetz der Kooperation überlegen.

Man darf allerdings bei diesem Beispiel nicht vergessen, dass es sich um ein sehr einfaches Modell mit einer sehr einfachen Netzfunktion handelt. Die affin lineare Regression erlaubt eine starke Extrapolationsfähigkeit, wird aber den meisten realen Problemen nicht gerecht. Auf der anderen Seite ist ein komplexeres Modell mit deutlich mehr Basisfunktionen als Eingängen kaum mehr anschaulich durch eine Analyse der Eigenvektoren von AB^{-1} zu beschreiben. In realen Anwendungen dürften die Extrapolationsfähigkeiten sowohl der kooperierenden Netze als auch des Gesamtnetzes deutlich schwächer ausfallen, was dann eher zu kleineren Quotienten $Q(x)$ führen dürfte.

Man darf auch nicht vergessen, dass hier genau genommen zwei Modelle mit unterschiedlicher Komplexität verglichen werden: während das Gesamtnetz drei Gewichte besitzt, verfügt die Kooperation über immerhin sechs Gewichte. Man bedenke auch, dass alle Gewichte nicht regularisiert sind.

Für die Strategie der Aufteilung der Trainingsdaten auf Netze gilt daher folgende vorläufige Empfehlung:

> Eine Menge von Trainingsdaten sollte nur dann auf mehrere Netze aufgeteilt werden, wenn ein Gesamtnetz aufgrund von zu hoher Komplexität in Speicher und/oder Rechenzeit nicht realisierbar ist.
> Die Aufteilung sollte dabei Clustern in den Trainingsdaten folgen.

Diese Empfehlung setzt natürlich a priori Wissen über die Verteilung möglicher Prognoseanfragen voraus, die aber in den meisten Fällen durch die Verteilung der Trainingsdaten approximiert wird.

Der Grenzfall der Aufteilung bestünde in einem eigenen Netz für jeden Trainingsdatenpunkt. Eine derartige Aufteilung ist aber nicht sinnvoll. Zwar würde sich die Rechen- und Speicherplatzkomplexität der Prognose durch dieses Verfahren von $O(N^2)$ beim Gesamtnetz zu $O(N)$ bei der Kooperation reduzieren, aber die Prognosen wären praktisch überall schlecht. Ein solches Verfahren entspräche einer verallgemeinerten Mittelwertbildung über die Trainingswerte, die bekanntermaßen keine gute Generalisierungsfähigkeit besitzt.

4.2.3 Einfluss der Gewichtsregularisierung

In Abschnitt 4.2.1 wurde gezeigt, dass $\sigma_K^2(x) \geq \sigma_{AB}^2(x)$ ist, falls die Gewichte a priori gleichverteilt sind. Bei praktischen Netzen wird jedoch eine Gewichtsregularisierung verwendet, die zu erweiterten Matrizen führt. Die optimale a priori Verteilung der Gewichte stellt (bei vorgegebenen Basisfunktionen) die Komplexität der zu lernenden Funktion dar, daher wird hier angenommen, dass alle Netze die gleiche Gewichtsregularisierung erfahren:

$$\sigma_A^2(x) = g(x)^T \left(\sigma_\mathrm{w}^{-2} I + A_A\right)^{-1} g(x) \tag{4.65}$$

$$\sigma_B^2(x) = g(x)^T \left(\sigma_\mathrm{w}^{-2} I + A_B\right)^{-1} g(x) \tag{4.66}$$

$$\sigma_K^2(x) = \left(\sigma_A^{-2}(x) + \sigma_B^{-2}(x)\right)^{-1} \tag{4.67}$$

$$\sigma_{AB}^2(x) = g(x)^T \left(\sigma_\mathrm{w}^{-2} I + A_A + A_B\right)^{-1} g(x). \tag{4.68}$$

Für $\sigma_\mathrm{w} \to \infty$ gilt wie oben gezeigt $\sigma_K^2(x) \geq \sigma_{AB}^2(x)$. Im Grenzfall $\sigma_\mathrm{w} \to 0$ gilt aber

$$\begin{aligned}
\lim_{\sigma_\mathrm{w}\to 0} \frac{\sigma_K^2(x)}{\sigma_{AB}^2(x)} &= \lim_{\sigma_\mathrm{w}\to 0} \frac{\left(\left(g(x)^T \left(\sigma_\mathrm{w}^{-2} I + A_A\right)^{-1} g(x)\right)^{-1} + \left(g(x)^T \left(\sigma_\mathrm{w}^{-2} I + A_B\right)^{-1} g(x)\right)^{-1}\right)^{-1}}{g(x)^T \left(\sigma_\mathrm{w}^{-2} I + A_A + A_B\right)^{-1} g(x)} \\
&= \lim_{\sigma_\mathrm{w}\to 0} \frac{\left(\left(g(x)^T \left(I + \sigma_\mathrm{w}^2 A_A\right)^{-1} g(x)\right)^{-1} + \left(g(x)^T \left(I + \sigma_\mathrm{w}^2 A_B\right)^{-1} g(x)\right)^{-1}\right)^{-1}}{g(x)^T \left(I + \sigma_\mathrm{w}^2 A_A + \sigma_\mathrm{w}^2 A_B\right)^{-1} g(x)} \\
&= \frac{\left(\left(g(x)^T g(x)\right)^{-1} + \left(g(x)^T g(x)\right)^{-1}\right)^{-1}}{g(x)^T g(x)} \\
&= \frac{1}{2}.
\end{aligned} \tag{4.69}$$

Falls also σ_{w} genügend klein ist, gilt also $\sigma_K^2(x) < \sigma_{AB}^2(x)$. Damit ist für ein allgemeines σ_{w}, also eine beim Training optimierte Gewichtsregularisierung, keine vergleichende Aussage zwischen den Größen $\sigma_K^2(x)$ und $\sigma_{AB}^2(x)$ möglich.

Zwar lässt eine reale Gewichtsregularisierung keine eindeutige Aussage mehr über das Verhältnis der Prognosefehler zu, aber die Prognosefehler verlieren in dieser Situation auch den Charakter eines Maßstabs für die Generalisierungsfähigkeit. Die Gewichtsregularisierung führt nämlich einerseits zu kleineren Prognosefehlern, gehört aber andererseits zum a priori Wissen, enthält also kein Datenwissen. Somit bleibt die Aussage „Die Kooperation kann nicht besser generalisieren als ein Gesamtnetz.“ in gewisser Weise auch bei realer Gewichtsregularisierung gültig.

4.3 Lernen diskontinuierlicher Ausgangsgrößen

[2]Die in Abschnitt 3.1 vorgestellten Netze (erweitert um die Methoden aus Abschnitt 3.2) sind in der Lage zu gegebenen reellen Messwerten mit angegebenen Messfehlern Prognosewerte und -fehler zu berechnen (Regression). Dieser Abschnitt beschreibt ergänzend dazu ein Verfahren zur Verarbeitung diskontinuierlicher Messgrößen (Klassifikation). Um das Gesamtsystem übersichtlich zu strukturieren und die Trainingsalgorithmen einfach zu gestalten, soll die Klassifikation auf die Regression zurückgeführt werden. Im Rahmen der Vorverarbeitung sollen dazu diskontinuierliche Parameter auf kontinuierliche abgebildet werden, die nach der Prognose dann entsprechend in diskontinuierliche Werte zurücktransformiert werden können.

Das hier vorgeschlagene Verfahren zur Behandlung von diskontinuierlichen Parametern unterscheidet sich von der in der Literatur üblichen Behandlung, etwa bei [Bidasaria], [Campbell], [Kulikowski], [MacKay3], [MixJon], [PosMar], [TitLik] und [Torrieri], grundsätzlich durch das zugrunde liegende Modell. Dort wird üblicherweise von folgendem Verfahren zur Gewinnung eines Datensatzes ausgegangen: zunächst wird bewusst eine Klasse C_n gewählt und anschließend der zugehörige Eingangsdatenvektor x_n ermittelt. Die Eingangsdatenvektoren einer Klasse sind Zufallsvariablen, wobei die Bestimmung ihrer Verteilung das eigentliche Ziel des Lernens ist (*class conditional density estimation*).

Als Beispiel sei ein Netz genannt, das anhand von verschiedenen messbaren physiologischen Eigenschaften das Risiko des Ausbruchs einer bestimmten Krankheit ermitteln soll. Die Trainingsdaten für dieses Netz werden üblicherweise wie folgt gesammelt: man wählt zunächst eine bestimmte Menge von Personen, von denen man weiß, dass sie an der Krankheit erkrankt sind (C_n =*krank*) und eine bestimmte Menge von Personen, von denen man weiß, dass sie gesund sind (C_n =*gesund*). Anschließend werden von all diesen Personen die benötigten physiologischen Eigenschaften ermittelt (x_n = (Gewicht, Alter, Blutdruck,...)).

In technischen Messdatensammlungen geht man üblicherweise den umgekehrten Weg. Es werden zunächst die Eingangsdaten für eine Messstelle x_n bestimmt und an der Messapparatur eingestellt. Die Eingangsdaten bestimmen sich nach dem Interesse des Experimentators, der in einem bestimmten Bereich neue Erkenntnisse in Form von Messdaten sammeln möchte. Im Laufe des Experiments werden dann bestimmte kontinuierliche Größen t_n (hier ein Vektor) gemessen, aus denen dann im Rahmen einer Auswertung diskontinuierliche Größen abgeleitet werden (*discrete-valued function estimation*). Bei perfekter Messung wäre die Bestimmung der diskontinuierlichen Größen deterministisch; da aber jede Messung einem Fehler unterliegt, ist auch die Bestimmung einer diskontinuierlichen Größe mit einer bestimmten Irrtumswahrscheinlichkeit behaftet.

Beispielhaft kann man hier die Frage, ob in einem Korrosionssystem Lochfraß auftritt, anführen. Gemäß DIN-Norm 50900 (siehe auch [Gräfen]) liegt Lochfraß genau dann vor, wenn die Tiefe der Vertiefungen der Werkstoffoberfläche größer ist als deren Durchmesser. Zunächst einmal wird der Experimentator ein Medium, einen Werkstoff und die Belastungsparameter auswählen und das Korrosionsexperiment durchführen. Anschließend wird er die Oberfläche des möglicherweise korrodierten Werkstoffs untersuchen und die geometrischen Maße der Vertiefungen bestimmen. Diese Maße unterliegen Messfehlern, sodass sich die Frage, ob Lochfraß vorliegt, nicht exakt, sondern nur in Form einer Wahrscheinlichkeitsaussage beantworten lässt. Sind die geometrischen Maße noch ermittelbar, kann die Wahrscheinlichkeit geschätzt werden: sind Durchmesser und Tiefe sehr verschieden, so ist ein Irrtum praktisch ausgeschlossen, sind

[2]Wesentliche Teile dieses Abschnitts wurden bereits in [Weber] vorveröffentlicht.

sie annähernd gleich, ist ein Irrtum nicht unwahrscheinlich. Falls die geometrischen Maße nicht mehr rekonstruiert werden können, muss von einer allgemeinen Irrtumswahrscheinlichkeit ausgegangen werden. Höhere Sicherheit in der Prognose kann dann nur durch benachbarte Messpunkte gleichen Verhaltens erreicht werden.

Die Strategie der Rückführung des diskontinuierlichen Problems auf ein Netz nach Abschnitt 3.1 ist stellen- bzw. punktweise. Während man bei Regressionsproblemen von der Lösung Stetigkeit und Dehnungsbeschränkung fordert, gibt es bei Klassifikationsproblemen keine vergleichbare Forderung. Daher wird im Folgenden von der Abhängigkeit der Messungen von einer Messstelle x_n abstrahiert und nur die Interaktion mehrerer Messungen an der gleichen Stelle betrachtet (die Variablen x_n und x werden folglich weggelassen). Erst die verwendeten Regressionsnetze verknüpfen wieder verschiedene Messstellen miteinander.

Der weitere Text dieses Abschnitts gliedert sich in drei Teile. Zunächst wird ein Zwei-Klassen-Problem diskutiert, das sich mit einem einzigen Netz nach Abschnitt 3.1 lösen lässt. Anschließend wird das allgemeine K-Klassen-Problem vorgestellt und auf K Netze nach Abschnitt 3.1 zurückgeführt. Der erste Teil ist dabei etwas verständlicher, seine Lösung ist aber kein Spezialfall des zweiten Teils. Der zweite Teil beschreibt die aktuelle Implementierung, der dritte Teil stellt empirische Auswertungen vor.

4.3.1 Ein Modell für zwei Klassen

Gegeben seien zwei Klassen C_1 und C_2, von denen immer genau eine auftritt bzw. beobachtet wird. An der betrachteten Stelle x wurden N stochastisch unabhängige Beobachtungen $\varphi_1, \ldots, \varphi_N$ gemacht. Die a priori Wahrscheinlichkeiten seien für beide Klassen gleich, während die Wahrscheinlichkeiten für die wahre Klasse bei gegebener beobachteter Klasse durch die Konstanten P_1 und P_2 gegeben seien:

$$f, \varphi_1, \ldots, \varphi_N \in \{C_1, C_2\} \tag{4.70}$$

$$P(f = C_1) = P(f = C_2) = \frac{1}{2} \tag{4.71}$$

$$P(f = C_1 | \varphi_n = C_1) = P_1 \tag{4.72}$$

$$P(f = C_2 | \varphi_n = C_2) = P_2 \tag{4.73}$$

$$\varphi_1 | f, \ldots, \varphi_N | f \quad \text{sind stochastisch unabhängig.} \tag{4.74}$$

Im Folgenden wird die Notation von f der Einfachheit halber weggelassen, wenn eine konkrete Klasse genannt ist, d.h. $P(C_1) := P(f = C_1)$.

Hat man nicht die bedingten Wahrscheinlichkeiten für die wahre Klasse $P(f|\varphi)$, sondern — analog zum kontinuierlichen Fall — die bedingten Wahrscheinlichkeiten für die beobachteten Klassen $P(\varphi|f)$, so kann man umrechnen:

$$\begin{aligned} P(f|\varphi) &= \frac{P(f)P(\varphi|f)}{\sum_{i=1}^{2} P(C_i)P(\varphi|C_i)} \\ &= \frac{P(\varphi|f)}{\sum_{i=1}^{2} P(\varphi|C_i)}. \end{aligned} \tag{4.75}$$

Diese Umrechnung wird dann nicht zur Laufzeit berechnet, sondern kann vorab bestimmt werden.

Die a posteriori Verteilung der wahren Klasse f bei gegebenen Messungen $\varphi_1, \ldots, \varphi_N$ an gleicher Stelle kann nun wie folgt bestimmt werden:

$$\begin{aligned} P(f|\varphi_1, \ldots, \varphi_N) &= \frac{P(f)P(\varphi_1, \ldots, \varphi_N|f)}{P(\varphi_1, \ldots, \varphi_N)} \\ &= \frac{P(f)P(\varphi_1, \ldots, \varphi_N|f)}{P(C_1)P(\varphi_1, \ldots, \varphi_N|C_1) + P(C_2)P(\varphi_1, \ldots, \varphi_N|C_2)} \\ &= \frac{P(f)\prod_{n=1}^{N} P(\varphi_n|f)}{P(C_1)\prod_{n=1}^{N} P(\varphi_n|C_1) + P(C_2)\prod_{n=1}^{N} P(\varphi_n|C_2)} \\ &= \frac{P(f)\prod_{n=1}^{N} \frac{P(\varphi_n)P(f|\varphi_n)}{P(f)}}{P(C_1)\prod_{n=1}^{N} \frac{P(\varphi_n)P(C_1|\varphi_n)}{P(C_1)} + P(C_2)\prod_{n=1}^{N} \frac{P(\varphi_n)P(C_2|\varphi_n)}{P(C_2)}} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\prod_{n=1}^{N} P(f|\varphi_n)}{\prod_{n=1}^{N} P(C_1|\varphi_n) + \prod_{n=1}^{N} P(C_2|\varphi_n)} \\
&= \frac{\prod_{n=1}^{N} P(f|\varphi_n)}{\prod_{n=1}^{N} P(f|\varphi_n) + \prod_{n=1}^{N} (1 - P(f|\varphi_n))} \\
&= \frac{1}{1 + \prod_{n=1}^{N} \left(\frac{1}{P(f|\varphi_n)} - 1 \right)}. \qquad (4.76)
\end{aligned}
$$

Hätte man tatsächlich nur eine einzige Messstelle, wäre das Problem durch Anwendung dieser Gleichung bereits gelöst.

Um verschiedene Messstellen miteinander interagieren zu lassen, wird ein (kontinuierliches) Netz verwendet. Dieses berechnet aus stochastisch unabhängigen Messwerten $t_1, \ldots, t_N$ und dazugehörigen Messfehlern $s_1, \ldots, s_N$ an einer Stelle x Prognosewerte μ und Prognosefehler σ. Das verwendete Netz kann ein Netz nach Abschnitt 3.1 sein. Wir abstrahieren hier aber von der konkreten Arbeitsweise der bayesschen Methoden und nehmen daher die Eigenschaften

$$t_1, \ldots, t_N \qquad \text{unabhängige Messwerte an gemeinsamer Stelle } x, \qquad (4.77)$$

$$s_1, \ldots, s_N \qquad \text{zugehörige Messfehler}, \qquad (4.78)$$

$$\sigma = \left(\sum_{n=1}^{N} s_n^{-2} \right)^{-1/2} \qquad \text{Prognosefehler}, \qquad (4.79)$$

$$\mu = \sigma^2 \sum_{n=1}^{N} s_n^{-2} t_n \qquad \text{Prognosewert}. \qquad (4.80)$$

als vereinfachendes Modell des Netzes an. Diese Gleichungen entsprechen den Näherungen 3.100 und 3.101.

Das Klassifikationsproblem soll nun durch ein Netz gelöst werden. Dazu wird eine Funktion h gesucht, die jedem Prognosewert μ und jedem Prognosefehler σ die a posteriori Wahrscheinlichkeit für das Ereignis $f = C_1$ zuordnet:

$$h(\mu, \sigma) := P(C_1|\varphi_1, \ldots, \varphi_N). \qquad (4.81)$$

Betrachtet man nur einen einzelnen Trainingsdatensatz, $N = 1$, so folgt aus den Gleichungen 4.79, 4.80 und 4.81 die Gleichung $h(t_1, s_1) = P(C_1|\varphi_1)$. Diese soll nun für jeden einzelnen Datensatz gelten:

$$\forall n = 1, \ldots, N: \quad h(t_n, s_n) = P(C_1|\varphi_n). \qquad (4.82)$$

Da es nur zwei verschiedene beobachtbare Klassen gibt, gibt es auch nur zwei verschiedene „Messwerte“ $t^{(1)}$ und $t^{(2)}$ mit zugehörigen „Messfehlern“ $s^{(1)}$ und $s^{(2)}$ in den Trainingsdaten des Netzes. Diese Werte seien hier zunächst vorgegeben, sie werden weiter unten konkret bestimmt. Die Transformation der beobachteten Klassen beim Training geschieht also durch die Abbildung

$$\varphi_n = C_1 \mapsto t_n = t^{(1)},\ s_n = s^{(1)} \qquad (4.83)$$

$$\varphi_n = C_2 \mapsto t_n = t^{(2)},\ s_n = s^{(2)} \qquad (4.84)$$

und die Rücktransformation bei der Prognose durch

$$\mu, \sigma \mapsto P(C_1|\varphi_1, \ldots, \varphi_N). \qquad (4.85)$$

Abbildung 4.2 stellt dies in einer Übersicht dar.

Aus diesem Abbildungsschema folgen nun einige Bedingungen an die Funktion h.

Bedingung I
$$h(t^{(1)}, s^{(1)}) = P_1 \qquad (4.86)$$

Bedingung II
$$h(t^{(2)}, s^{(2)}) = 1 - P_2 \qquad (4.87)$$

Bedingung III
$$h\left(\frac{\sum_{n=1}^{N} s_n^{-2} t_n}{\sum_{n=1}^{N} s_n^{-2}}, \frac{1}{\sqrt{\sum_{n=1}^{N} s_n^{-2}}} \right) = \frac{1}{1 + \prod_{n=1}^{N} \left(\frac{1}{h(t_n, s_n)} - 1 \right)} \qquad (4.88)$$

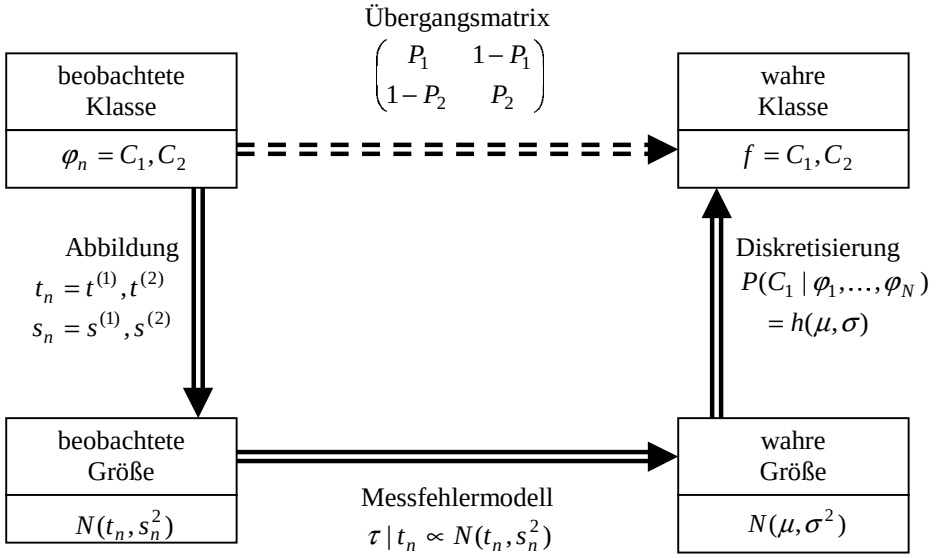

Abbildung 4.2: Schematische Darstellung der Abbildung und Berechnung der diskontinuierlichen Prognosen. Links befinden sich die Beobachtungen und Trainingsdaten, rechts die Prognosen. Oben befinden sich die diskontinuierlichen Größen und deren Wahrscheinlichkeitsangaben, unten die kontinuierlichen Größen. Man beachte, dass eine direkte Berechnung entlang des gestrichelten Pfeils nicht möglich ist, da eine Generalisierung über verschiedene Messstellen dort nicht in dieser Architektur möglich ist.

Zunächst einmal soll die Rücktransformationsfunktion $h(.,.)$ bei genau einer Messung deren Klassenwahrscheinlichkeit reproduzieren: die Bedingungen I und II folgen aus den Gleichungen 4.72, 4.73 4.82, 4.83 und 4.84. Weiter soll sie gemäß den Gleichungen 4.76, 4.79, 4.80, 4.81 und 4.82 mehrere Messungen in den kontinuierlichen Größen so zusammenfassen, dass dies der Zusammenfassung der diskontinuierlichen Größen entspricht (Bedingung III). Gemäß Lemma 1 in Anhang B erfüllt die folgende Wahl von h die Bedingungen I, II und III:

$$h(t,s) = \frac{1}{1+\exp\left(-\alpha\frac{t-T}{s^2}\right)} \quad \text{mit} \tag{4.89}$$

$$\alpha := \frac{c_1 - c_2}{t^{(1)} - t^{(2)}} \tag{4.90}$$

$$T := \frac{c_1 t^{(2)} - c_2 t^{(1)}}{c_1 - c_2} \tag{4.91}$$

$$c_1 := (s^{(1)})^2 \ln\left(\frac{P_1}{1-P_1}\right) \tag{4.92}$$

$$c_2 := (s^{(2)})^2 \ln\left(\frac{1-P_2}{P_2}\right). \tag{4.93}$$

Bis hierher wurden die Parameter $t^{(1)},\ t^{(2)} \in \mathbb{R}$ und $s^{(1)},\ s^{(2)} \in \mathbb{R}^+$ als fest vorgegeben betrachtet. Es wurde gezeigt, dass man für jede beliebige Wahl dieser vier Konstanten, die die Ungleichungen $t^{(1)} \neq t^{(2)}$ und $(s^{(1)})^2 \ln\left(\frac{P_1}{1-P_1}\right) \neq (s^{(2)})^2 \ln\left(\frac{1-P_2}{P_2}\right)$ erfüllen[3], immer eine Funktion h finden kann, die den Bedingungen I bis III genügt. Dies bedeutet, dass es bei den bisherigen Betrachtungen keine Rolle gespielt hat, ob man dem Netz sehr präzise Messwerte ($s^{(1)}, s^{(2)} \ll |t^{(1)} - t^{(2)}|$), sehr unpräzise Messwerte ($s^{(1)}, s^{(2)} \gg |t^{(1)} - t^{(2)}|$) oder sogar einen sehr präzisen und einen sehr unpräzisen Messwert

[3]Beide Ungleichungen sind in der Praxis immer erfüllt. Die erste drückt lediglich aus, dass bei unterschiedlichen beobachteten Klassen auch unterschiedliche Trainingswerte gelernt werden. Für eine sinnvolle Anwendung sollten immer $P_1 > 0,5$ und $P_2 > 0,5$ sein, was die Erfüllung der zweiten Ungleichung garantiert.

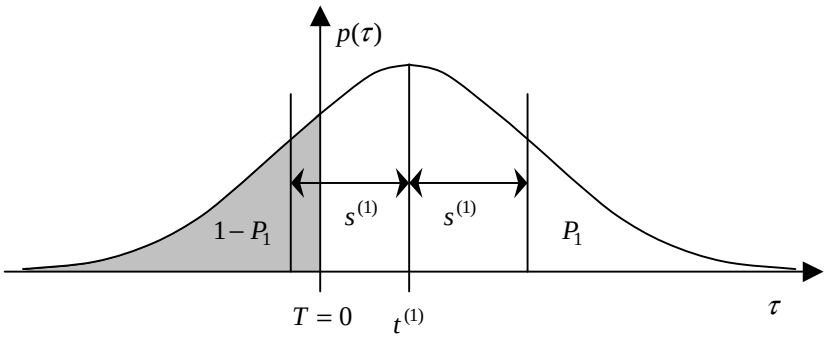

Abbildung 4.3: Illustration der Verteilung der Netzausgabe τ unter nur einem Trainingsdatensatz $(t^{(1)}, s^{(1)})$.

$(s^{(1)} \ll |t^{(1)} - t^{(2)}| \ll s^{(2)})$ präsentiert hat. Die Wahl der Konstanten hat — im Rahmen der Annahmen über das Netz — keine Auswirkungen auf die a posteriori Wahrscheinlichkeitsaussage $P(C_1|\varphi_1, \ldots, \varphi_N)$ nach Anwendung von h.

Allerdings sind verschiedene Eigenschaften des verwendeten Netzes sehr wesentlich von seinen Trainingsdaten abhängig: so etwa das Verhältnis zwischen den Trainingswerten und den a priori Prognosen[4] oder die Evidenz des Netzes. Werden neben der diskontinuierlichen Größe auch andere (z.B. kontinuierliche) Größen trainiert, und verwendet das Netz für seine Ausgänge gemeinsame Regularisierungen (gemeinsame Parameter der a priori Verteilung von Gewichten, die unterschiedlichen Ausgängen zugeordnet sind), so muss die Genauigkeit, mit der die Messwerte der einzelnen Ausgänge approximiert werden sollen, vergleichbar gemacht werden.

Um hier Abhilfe zu schaffen, werden Bedingungen für die Konstanten eingeführt. Da die a priori Prognosen bei vielen Implementierungen von Netzen den Erwartungswert Null besitzen und auch die a posteriori Prognosen aufgrund der Gewichtsregularisierung in Richtung Null verzerrt sind (Abschnitt 3.3.5), sollen für den Prognosewert $\mu = 0$ immer beide Klassen C_1 und C_2 gleichwahrscheinlich sein. Dies bedeutet

$$\begin{aligned} & \forall \sigma \in \mathbb{R}^+ : \quad h(0, \sigma) = \frac{1}{2} \\ \iff \quad & \forall \sigma \in \mathbb{R}^+ : \quad \frac{1}{1 + \exp\left(-\alpha \frac{-T}{\sigma^2}\right)} = \frac{1}{2} \\ \iff \quad & T = 0, \end{aligned} \tag{4.94}$$

da $\alpha \neq 0$ sein muss. Weil von den Netzen Invarianz bezüglich der linearen Transformation der Ausgangsgrößen (Wert gemeinsam mit Fehler) erwartet wird, ist die Wahl von α für die verwendeten Netze mit nur je einem Ausgang willkürlich. Es sei daher

$$\alpha \quad = \quad 1. \tag{4.95}$$

Zwei weitere Bedingungen für die Konstanten ergeben sich durch eine Skalierung der Ausgangsverteilung des Netzes. Die Prognose des Netzes ist ursprünglich eine Zufallsvariable $\tau \propto \mathcal{N}\left(\mu, \sigma^2\right)$, die die geschätzte Verteilung des wahren Werts angibt. Wüsste man den wahren Wert für τ, dann wäre — so unser Modell — C_1 genau dann die wahre Klasse, wenn $h(\tau, \sigma) > 0,5$ ist. Dieses Argument soll nun auf die Trainingsdaten angewendet werden, siehe dazu Abbildung 4.3. Für die Klasse C_1 mit den Trainingsdaten $(t^{(1)},\ s^{(1)})$ soll gelten:

$$P_1 \ = \ P(C_1|\varphi_n = C_1) \quad = \quad P\left(h(\tau, s^{(1)}) > 0,5 \mid \tau \propto \mathcal{N}\left(t^{(1)}, (s^{(1)})^2\right)\right) \tag{4.96}$$

[4]Die Prognosen eines Netzes, das mit 0 Trainingsdaten „trainiert“ wurde. Die Gewichte sind dann a priori verteilt und implizieren auch so eine Verteilung der Netzausgangsvariablen $t|x$, siehe auch Abschnitt 3.3.6.

$$
\begin{aligned}
&= P\left(\frac{1}{1+\exp\left(-\alpha\frac{\tau-T}{(s^{(1)})^2}\right)} > 0,5 \mid \tau \propto \mathcal{N}\left(t^{(1)}, (s^{(1)})^2\right)\right) \\
&= P\left(\tau > T \mid \tau \propto \mathcal{N}\left(t^{(1)}, (s^{(1)})^2\right)\right) \\
&= P\left(s^{(1)}\tau + t^{(1)} > T \mid \tau \propto \mathcal{N}(0,1)\right) \\
&= P\left(-\tau < \frac{t^{(1)}-T}{s^{(1)}} \mid \tau \propto \mathcal{N}(0,1)\right) \\
&= P\left(\tau < \frac{t^{(1)}-T}{s^{(1)}} \mid \tau \propto \mathcal{N}(0,1)\right) \\
&= \phi\left(\frac{t^{(1)}-T}{s^{(1)}}\right), \qquad (4.97)
\end{aligned}
$$

wobei die Funktion $\phi : \mathbb{R} \to]0,1[$ die Verteilungsfunktion der Standardnormalverteilung, gegeben durch

$$\phi(t) := \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) dz, \qquad (4.98)$$

ist. Diese Normierungsbedingung führt zu stabilem Verhalten bei der Rücktransformation der Prognosen, wenn das Netz verzerrte Prognosen liefert. Die Größe der Verzerrung ist dabei durch den Prognosefehler σ gegeben, ihre Ursache kann dabei etwa eine ungünstige Netzfunktion (z.B. ungleichmäßig verteilte Basisfunktionen) sein. Analog zu Gleichung 4.97 gelten die gleichen Überlegungen auch für den Punkt $(t^{(2)},\ s^{(2)})$, woraus

$$P_2 = \phi\left(\frac{t^{(2)}-T}{s^{(2)}}\right), \qquad (4.99)$$

folgt.

Mit den Gleichungen 4.94, 4.95, 4.97 und 4.99 sind nun vier Bedingungen für vier Konstanten definiert worden. Die eindeutig bestimmte Lösung dieser Gleichungen lautet

$$t^{(1)} = \frac{\left(\phi^{-1}(P_1)\right)^2}{\ln\frac{P_1}{1-P_1}} \qquad (4.100)$$

$$t^{(2)} = \frac{\left(\phi^{-1}(P_2)\right)^2}{\ln\frac{P_2}{1-P_2}} \qquad (4.101)$$

$$s^{(1)} = \frac{\phi^{-1}(P_1)}{\ln\frac{P_1}{1-P_1}} \qquad (4.102)$$

$$s^{(2)} = \frac{\phi^{-1}(P_2)}{\ln\frac{P_2}{1-P_2}}, \qquad (4.103)$$

wobei ϕ^{-1} die Umkehrfunktion von ϕ ist. Abbildung 4.4 stellt die Abhängigkeit der Konstanten $t^{(1)}$ und $s^{(1)}$ von der gegebenen Wahrscheinlichkeit P_1 dar. Die Funktion ϕ kann effizient über eine Kettenbruchentwicklung berechnet werden [Müller], die Funktion ϕ^{-1} darauf aufbauend über eine Newton-Nullstellensuche.

4.3.2 Ein Modell für mehrere Klassen

Wir betrachten nun eine wahre Funktion $f : \mathbb{R}^L \to \{C_1, \ldots, C_K\}$, deren Wert eine von mehreren Klassen ist. Diese Funktion wurde N-mal gemessen (wir betrachten hier wieder nur eine Stelle x), bei jeder Messung wurde eine von J Hinweisklassen $D_1, \ldots, D_J$ beobachtet. Die beiden Klassenmengen sind durch ihre Korrelationsmatrix verbunden, die bekannt und Teil des konkreten Modells ist: für jede mögliche wahre Klasse C_k und jede beobachtbare Klasse D_j sei die Wahrscheinlichkeit $P(C_k|D_j)$ vorgegeben.

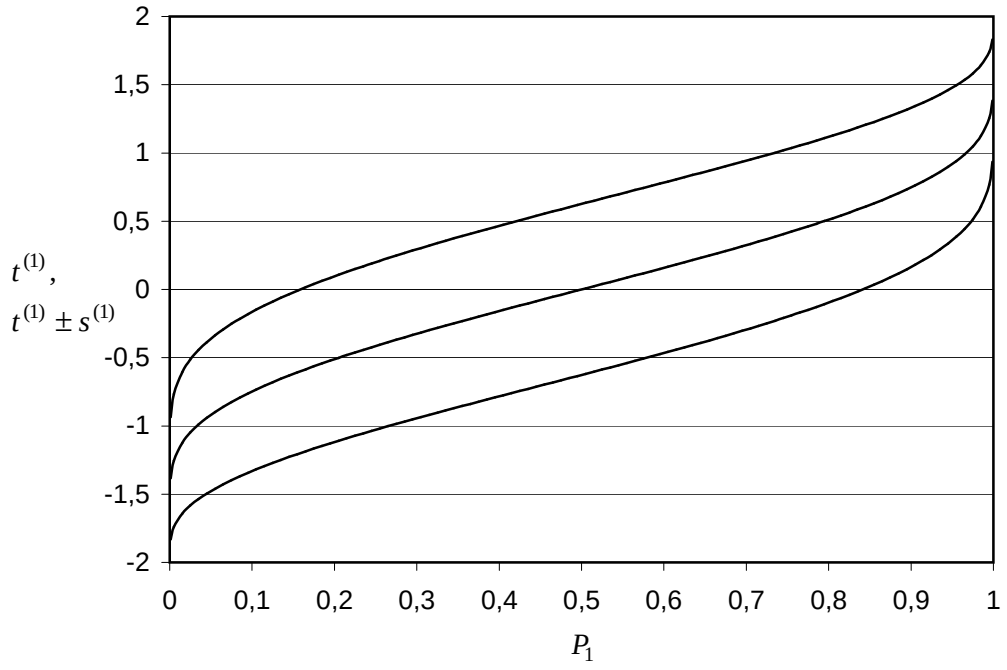

Abbildung 4.4: Abhängigkeit der Trainingswerte und -fehler von der Wahrscheinlichkeit P_1 bei Beobachtung von C_1. P_1 auf der Abszisse ist die gegebene Wahrscheinlichkeit dafür, dass C_1 auch die wahre Klasse ist. Auf der Ordinate ist der jeweilige Trainingswert $t^{(1)}$ (mittlere Kurve) zusammen mit dem einfachen Trainingsfehlerintervall $t^{(1)} \pm s^{(1)}$ (obere und untere Kurve) dargestellt.

An dieser Stelle sei bemerkt, dass die Menge der möglichen wahren Klassen, $\{C_1, \ldots, C_K\}$, und die Menge der beobachtbaren Klassen, $\{D_1, \ldots, D_J\}$, gleich sein können, aber nicht müssen. Ein potenzieller Grund für eine Ungleichheit ist etwa die Zusammenfassung mehrerer beobachtbarer Klassen zu einer möglichen wahren Klasse. Die zu einer wahren Klasse C_k gehörenden beobachtbaren Klassen $D_{j1}, D_{j2}, \ldots$ können sich dann durch die Wahrscheinlichkeiten $P(C_k|D_{j1}), P(C_k|D_{j2}), \ldots$ der richtigen Klassifikation und $P(C_{k'}|D_{j1}), P(C_{k'}|D_{j2}), \ldots$ einer Fehlklassifikation (Falschmessung) in Klasse $C_{k'}$ unterscheiden. Beispielhaft sei etwa für den Parameter Korrosionsart[5] die Menge der beobachtbaren Klassen die Menge {*gleichmäßig, muldenförmig, lochförmig*}. Die Menge der wahren Klassen sei daraus gebildet, indem *muldenförmig* und *lochförmig* zu *ungleichmäßig* zusammengefasst werden. Da *muldenförmig* inhaltlich irgendwo zwischen *gleichmäßig* und *lochförmig* steht, ist eine Fehlklassifikation bei der Beobachtung *muldenförmig* deutlich wahrscheinlicher als bei der Beobachtung *lochförmig*, obwohl beide mit hoher Wahrscheinlichkeit der wahren Klasse *ungleichmäßig* entstammen.

Das beschriebene Modell soll jetzt formalisiert werden. Die a priori Wahrscheinlichkeiten für die möglichen wahren Klassen seien für alle diese Klassen gleich groß. Die stochastisch unabhängigen Beobachtungen sollen mit $\varphi_1, \ldots, \varphi_N$ bezeichnet werden.

$$f \in \{C_1, \ldots, C_K\} \tag{4.104}$$

$$\varphi_1, \ldots, \varphi_N \in \{D_1, \ldots, D_J\} \tag{4.105}$$

$$P(f = C_1) = \ldots = P(f = C_K) = \frac{1}{K} \tag{4.106}$$

$$P_{kj} := P(f = C_k|\varphi_n = D_j) \tag{4.107}$$

$$= P(C_k|D_j)$$

$$\varphi_1|f, \ldots, \varphi_N|f \quad \text{sind stochastisch unabhängig.} \tag{4.108}$$

[5]In der vorliegenden Implementierung wurde der Parameter Korrosionsart zugunsten der Korrosionserscheinung nicht verwendet. Die Modellierung der Korrosionserscheinung ist aber erheblich komplexer.

Die Schätzung des wahren Werts f bei gegebenen Beobachtungen $\varphi_1, \ldots, \varphi_N$ (an gleicher Stelle) ergibt sich nun zu:

$$\begin{aligned} P(f|\varphi_1,\ldots,\varphi_N) &= \frac{P(f)P(\varphi_1,\ldots,\varphi_N|f)}{P(\varphi_1,\ldots,\varphi_N)} \\ &= \frac{P(f)P(\varphi_1,\ldots,\varphi_N|f)}{\sum_{i=1}^{K} P(C_i)P(\varphi_1,\ldots,\varphi_N|C_i)} \\ &= \frac{P(f)\prod_{n=1}^{N} P(\varphi_n|f)}{\sum_{i=1}^{K} P(C_i)\prod_{n=1}^{N} P(\varphi_n|C_i)} \\ &= \frac{P(f)\prod_{n=1}^{N} \frac{P(\varphi_n)P(f|\varphi_n)}{P(f)}}{\sum_{i=1}^{K} P(C_i)\prod_{n=1}^{N} \frac{P(\varphi_n)P(C_i|\varphi_n)}{P(C_i)}} \\ &= \frac{\prod_{n=1}^{N} P(f|\varphi_n)}{\sum_{i=1}^{K}\prod_{n=1}^{N} P(C_i|\varphi_n)}. \end{aligned} \tag{4.109}$$

Um das diskontinuierliche Problem zu lösen, steht ein (kontinuierliches) Netz mit K Ausgängen (z.B. K Netze nach Abschnitt 3.1) zur Verfügung. Der n-te Trainingsdatensatz für dieses Netz besteht aus einem Vektor von Trainingswerten für jeden Ausgang $t_n = (t_{n1}, \ldots, t_{nK})^T$ und einem Vektor von zugehörigen Trainingsfehlern $s_n = (s_{n1}, \ldots, s_{nK})^T$. Für jeden dieser Ausgänge verhält sich das Netz so wie ein ganzes Netz aus Abschnitt 4.3.1. Es gilt entsprechend der Näherungen 3.100 und 3.101 für Netze:

$$t_1, \ldots, t_N \quad \text{unabhängige Messwertvektoren an gemeinsamer Stelle } x, \tag{4.110}$$

$$s_1, \ldots, s_N \quad \text{zugehörige Messfehlervektoren,} \tag{4.111}$$

$$\forall k = 1,\ldots,K: \quad \sigma_k = \left(\sum_{n=1}^{N} s_{nk}^{-2}\right)^{-1/2} \quad \text{Prognosefehler des } k\text{-ten Ausgangs,} \tag{4.112}$$

$$\forall k = 1,\ldots,K: \quad \mu_k = \sigma_k^2 \sum_{n=1}^{N} s_{nk}^{-2} t_{nk} \quad \text{Prognosewert des } k\text{-ten Ausgangs.} \tag{4.113}$$

Das weitere Vorgehen ist analog zum Zwei-Klassen-Fall. Das diskontinuierliche Problem soll auf ein kontinuierliches Problem abgebildet und so gelöst werden. Dazu wird eine Abbildungsfunktion $h(\mu, \sigma, k)$ mit $\mu = (\mu_1, \ldots, \mu_K)^T \in \mathbb{R}^K$, $\sigma = (\sigma_1, \ldots, \sigma_K)^T \in (\mathbb{R}^+)^K$ und $k \in \{1, \ldots, K\}$ gesucht, die die a posteriori Wahrscheinlichkeit für die wahre Klasse C_k aufgrund des kontinuierlichen Prognosewertvektors μ und dessen Prognosefehlervektors σ berechnet.

$$h(\mu, \sigma, k) := P(C_k|\varphi_1, \ldots, \varphi_N) \tag{4.114}$$

Das Einsetzen der Gleichungen 4.112 und 4.113 für den Fall $N = 1$ ergibt die entsprechenden Aussagen für die Transformation der Beobachtungen in die kontinuierlichen Trainingsdaten:

$$\forall n = 1,\ldots,N: \quad h(t_n, s_n, k) = P(C_k|\varphi_n). \tag{4.115}$$

Zu bestimmen sind nun neben der Funktion h dazu passend zu jeder beobachtbaren Klasse D_j die Trainingsdatenvektoren $t^{(j)} = (t_1^{(j)}, \ldots, t_K^{(j)})^T$ und Trainingsfehlervektoren $s^{(j)} = (s_1^{(j)}, \ldots, s_K^{(j)})^T$. Diese stellen die kontinuierlichen Abbilde der Beobachtungen nach folgendem Schema dar:

$$\forall n = 1,\ldots,N: \quad \varphi_n = D_j \quad \mapsto \quad t_n = t^{(j)},\ s_n = s^{(j)}. \tag{4.116}$$

Aus den Gleichungen 4.107 bis 4.116 lassen sich nun die folgenden Bedingungen I und II ableiten. Die Bedingungen IIIa–c stellen Alternativen dar und werden zur eindeutigen Bestimmung aller Variablen

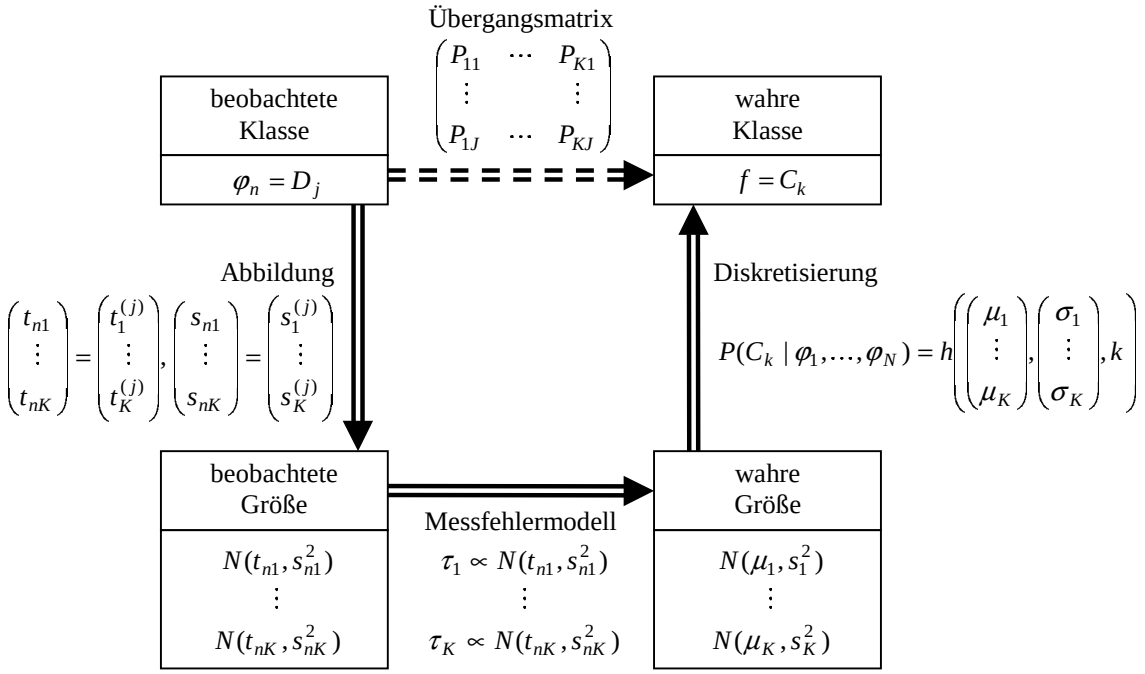

Abbildung 4.5: Schematische Darstellung der Abbildung und Berechnung der diskontinuierlichen Prognosen. In Analogie zu Abbildung 4.2 hier für den Mehrklassen-Fall.

benötigt.

Bedingung I $\forall j = 1, \ldots, J; k = 1, \ldots, K:$

$$h(t^{(j)}, s^{(j)}, k) = P_{kj} \tag{4.117}$$

Bedingung II $\forall t_1, \ldots, t_N, s_1, \ldots, s_N:$

$$h\left(\left(\frac{\sum_{n=1}^N s_{n1}^{-2} t_{n1}}{\sum_{n=1}^N s_{n1}^{-2}}, \ldots, \frac{\sum_{n=1}^N s_{nK}^{-2} t_{nK}}{\sum_{n=1}^N s_{nK}^{-2}}\right)^T, \left(\left(\sum_{n=1}^N s_{n1}^{-2}\right)^{-\frac{1}{2}}, \ldots, \left(\sum_{n=1}^N s_{nK}^{-2}\right)^{-\frac{1}{2}}\right)^T, k\right)$$

$$= \frac{\prod_{n=1}^N h(t_n, s_n, k)}{\sum_{i=1}^K \prod_{n=1}^N h(t_n, s_n, i)} \tag{4.118}$$

Bedingung IIIa $\forall j = 1, \ldots, J; k = 1, \ldots, K:$

$$P\left(\forall i \neq k: \ \tau_k > \tau_i \mid \forall i = 1, \ldots, K: \ \tau_i \propto \mathcal{N}\left(t_i^{(j)}, (s_i^{(j)})^2\right)\right) = P_{kj} \tag{4.119}$$

Bedingung IIIb $\forall j = 1, \ldots, J; k = 1, \ldots, K:$

$$P\left(\tau_k > \theta_j \mid \tau_k \propto \mathcal{N}\left(t_k^{(j)}, (s_k^{(j)})^2\right)\right) = P_{kj} \tag{4.120}$$

Bedingung IIIc $\forall j = 1, \ldots, J:$

$$s_1^{(j)} = \ldots = s_K^{(j)} =: s^{(j)} \tag{4.121}$$

Bedingung I ergibt sich direkt aus der Definition von h und den Trainingsdaten $t^{(j)}$ und $s^{(j)}$ und fordert, dass ein einzelner Datenpunkt vom System reproduziert wird. Bedingung II stellt sicher, dass sich das System mit einzelnen Messpunkten trainiert (rechte Seite) genauso verhält, wie wenn zunächst beobachtete Werte an einer Stelle zusammenfasst werden und das Netz dann mit den zusammengefassten Daten (linke Seite) trainiert wird.

Die Bedingungen IIIa–c sind als alternative Normierungen der Fehlerfunktion des Netzes zu verstehen und verhalten sich ähnlich wie Gleichung 4.96. Das Netz darf eine Abweichung zwischen Trainings- und Prognosewerten zulassen, wobei die Verteilung der Abweichung durch die Prognosefehler bestimmt wird; dies ist eine Folge seiner Generalisierungsfähigkeit. Es wird weiter angenommen, dass ein Prognosewertvektor μ des Netzes auf die Klasse C_k abgebildet wird, wenn $\mu_k > \mu_i$ für alle $i \neq k$ ist. Bedingung

IIIa fordert nun die Gleichheit zwischen der Wahrscheinlichkeit, dass das Netz aufgrund der genannten Abweichungen immer noch die Klasse C_k prognostiziert, und der Wahrscheinlichkeit, dass die Klasse C_k direkt durch die Messung als die wahre klassifiziert wird. Leider ist dieser Ansatz zwar ein gutes Modell, führt aber zu Abhängigkeiten der einzelnen Komponenten der Vektoren $t^{(j)}$ und $s^{(j)}$ und ist mathematisch schwer zu lösen (ggf. nichtexistente oder mehrdeutige Lösungen). Daher wird von einer weiteren Betrachtung abgesehen.

Bedingung IIIb beschreibt eine Näherung von Bedingung IIIa. Es wird angenommen, dass ein Prognosewert t_k des Netzes genau dann auf die vermutete wahre Klasse C_k abgebildet wird, wenn t_k einen bestimmten Schwellwert θ_j überschreitet. Der Schwellwert ersetzt also den Vergleich mit den anderen Komponenten von t und entkoppelt so die Variablen. Bedingung IIIb fordert nun, dass die Wahrscheinlichkeit, dass der k-te Ausgang des Netzes den Schwellwert θ_j überschreitet genauso groß sein soll wie die Wahrscheinlichkeit, dass die Klasse C_k direkt durch die Messung als die wahre klassifiziert wird.

Bedingung IIIc beschreibt einen völlig anderen Ansatz: hier steht nicht das Modell, sondern eine effiziente Implementierung im Vordergrund. Empirische Untersuchungen (Abschnitt 4.3.3) haben ergeben, dass die Gewichte, die mit den K Ausgängen verbunden sind, alle gleich regularisiert werden sollten, also gleiche Werte für σ_w verwenden sollten. Verwendet man generalisierte lineare Netze, dann sind unter Bedingung IIIc die Hesse-Matrizen (Gleichung 3.8) und somit auch die Prognosefehler (Gleichung 3.14) für alle K Ausgänge gleich, denn sie hängen nur von den Trainingsstellen, den Trainingsfehlern und der Gewichtsregularisierung ab. Gegenüber Bedingung IIIb kann man also (asymptotisch für $M \gg K$) den Faktor K an Speicherplatz und Rechenzeit bei der Prognose sparen, und auch das Training wird beschleunigt.

Die vorliegende Implementierung verwendet Bedingung IIIc. Da aber das System der Bedingungen I/II/IIIc die Trainingswerte und -fehler nicht eindeutig bestimmt, werden einige Konstanten in Anlehnung an die Lösung unter Bedingung IIIb gewählt. Daher werden hier beide Wege vollständig beschrieben.

Nun sollen für die Systeme der Bedingungen I/II/IIIb und I/II/IIIc konkrete Lösungen angegeben werden. Für beide Systeme gibt es eine gemeinsame Funktion

$$h(t,s,k) = \frac{\exp(s_k^{-2} t_k)}{\sum_{i=1}^{K} \exp(s_i^{-2} t_i)}, \tag{4.122}$$

die Bedingung II erfüllt, was wie folgt eingesehen werden kann. Die linke Seite von Bedingung II ergibt sich zu

$$\begin{aligned}
h\left(\left(\frac{\sum_{n=1}^{N} s_{n1}^{-2} t_{n1}}{\sum_{n=1}^{N} s_{n1}^{-2}}, \ldots, \frac{\sum_{n=1}^{N} s_{nK}^{-2} t_{nK}}{\sum_{n=1}^{N} s_{nK}^{-2}}\right)^T, \left(\left(\sum_{n=1}^{N} s_{n1}^{-2}\right)^{-\frac{1}{2}}, \ldots, \left(\sum_{n=1}^{N} s_{nK}^{-2}\right)^{-\frac{1}{2}}\right)^T, k\right) \\
&= \frac{\exp\left(\left(\sum_{n=1}^{N} s_{nk}^{-2}\right) \frac{\sum_{n=1}^{N} s_{nk}^{-2} t_{nk}}{\sum_{n=1}^{N} s_{nk}^{-2}}\right)}{\sum_{i=1}^{K} \exp\left(\left(\sum_{n=1}^{N} s_{ni}^{-2}\right) \frac{\sum_{n=1}^{N} s_{ni}^{-2} t_{ni}}{\sum_{n=1}^{N} s_{ni}^{-2}}\right)} \\
&= \frac{\exp\left(\sum_{n=1}^{N} s_{nk}^{-2} t_{nk}\right)}{\sum_{i=1}^{K} \exp\left(\sum_{n=1}^{N} s_{ni}^{-2} t_{ni}\right)},
\end{aligned} \tag{4.123}$$

während sich die rechte Seite wie folgt ergibt:

$$\frac{\prod_{n=1}^{N} h(t_n, s_n, k)}{\sum_{i=1}^{K} \prod_{n=1}^{N} h(t_n, s_n, i)} = \frac{\prod_{n=1}^{N} \frac{\exp(s_{nk}^{-2} t_{nk})}{\sum_{p=1}^{K} \exp(s_{np}^{-2} t_{np})}}{\sum_{i=1}^{K} \prod_{n=1}^{N} \frac{\exp(s_{ni}^{-2} t_{ni})}{\sum_{p=1}^{K} \exp(s_{np}^{-2} t_{np})}}$$

$$
\begin{aligned}
&= \frac{\prod_{n=1}^{N} \exp(s_{nk}^{-2} t_{nk})}{\sum_{i=1}^{K} \prod_{n=1}^{N} \exp(s_{ni}^{-2} t_{ni})} \\
&= \frac{\exp\left(\sum_{n=1}^{N} s_{nk}^{-2} t_{nk}\right)}{\sum_{i=1}^{K} \exp\left(\sum_{n=1}^{N} s_{ni}^{-2} t_{ni}\right)}. \qquad (4.124)
\end{aligned}
$$

Zur vollständigen Lösung der Abbildung müssen nun noch die Konstanten $t^{(j)}$, $s^{(j)}$ und θ_j für $j = 1, \ldots, J$ bestimmt werden. Einsetzen von Gleichung 4.122 in Bedingung I ergibt

$$
\begin{aligned}
\frac{\exp\left((s_k^{(j)})^{-2} t_k^{(j)}\right)}{\sum_{i=1}^{K} \exp\left((s_i^{(j)})^{-2} t_i^{(j)}\right)} &= P_{kj} \\
\exp\left((s_k^{(j)})^{-2} t_k^{(j)}\right) &= \underbrace{\left(\sum_{i=1}^{K} \exp\left((s_i^{(j)})^{-2} t_i^{(j)}\right)\right)}_{c_j} P_{kj} \\
(s_k^{(j)})^{-2} t_k^{(j)} &= \ln(c_j P_{kj}) \qquad (4.125)
\end{aligned}
$$

mit neuen Konstanten $c_1, \ldots, c_J \in \mathbb{R}^+$, die nun ebenfalls bestimmt werden müssen. Aus Bedingung IIIb folgt

$$
\begin{aligned}
P_{kj} &= P\left(\tau_k > \theta_j \mid \tau_k \propto \mathcal{N}\left(t_k^{(j)}, (s_k^{(j)})^2\right)\right) \\
&= P\left(s_k^{(j)} \tau_k + t_k^{(j)} > \theta_j \mid \tau_k \propto \mathcal{N}(0,1)\right) \\
&= P\left(\tau_k > (s_k^{(j)})^{-1}(\theta_j - t_k^{(j)}) \mid \tau_k \propto \mathcal{N}(0,1)\right) \\
&= 1 - \phi\left((s_k^{(j)})^{-1}(\theta_j - t_k^{(j)})\right) \\
&= \phi\left((s_k^{(j)})^{-1}(t_k^{(j)} - \theta_j)\right) \qquad (4.126)
\end{aligned}
$$

und weiter

$$
(s_k^{(j)})^{-1}(t_k^{(j)} - \theta_j) = \phi^{-1}(P_{kj}). \qquad (4.127)
$$

Die Gleichungen 4.125 und 4.127 bilden ein quadratisches Gleichungssystem, das die Variablen $t^{(j)}$ und $s^{(j)}$ für $j = 1, \ldots, J$ determiniert. Es fällt auf, dass die Lösungen für einzelne Werte von j unabhängig voneinander sind. Anschaulich bedeutet dies, dass jede einzelne Beobachtung für sich in Trainingswerte abgebildet werden kann ohne von anderen Beobachtungen abzuhängen. Für $\theta_j \neq 0$ ergeben sich die Lösungen des quadratischen Gleichungssystems 4.125 und 4.127 zu

$$
t_k^{(j)} = \theta_j + \frac{2\theta_j}{-1 \pm \sqrt{1 + 4\theta_j \frac{\ln(c_j P_{kj})}{(\phi^{-1}(P_{kj}))^2}}} \qquad (4.128)
$$

$$
s_k^{(j)} = \frac{2\theta_j}{\phi^{-1}(P_{kj})\left(-1 \pm \sqrt{1 + 4\theta_j \frac{\ln(c_j P_{kj})}{(\phi^{-1}(P_{kj}))^2}}\right)}. \qquad (4.129)
$$

Diese Lösungen sind aber nicht immer gültig. Zunächst müssen die Wurzeln auf den rechten Seiten definiert sein. Notwendige Bedingung dazu ist

$$\begin{aligned} 1 + 4\theta_j \frac{\ln(c_j P_{kj})}{\left(\phi^{-1}(P_{kj})\right)^2} &\geq 0 \\ \theta_j \frac{\ln(c_j P_{kj})}{\left(\phi^{-1}(P_{kj})\right)^2} &\geq -\frac{1}{4}. \end{aligned} \tag{4.130}$$

Man beachte, dass P_{kj} vorgegeben und von k echt abhängig ist, die zu wählenden Parameter c_j und θ_j sind dagegen von k unabhängig. Insbesondere kann der Zähler des Bruchs sowohl positiv als auch negativ werden während der Nenner nicht negativ ist. Daher wird

$$\theta_j = 0 \qquad \text{für} \qquad j = 1, \ldots, J \tag{4.131}$$

gewählt, womit dann die obige Ungleichung für alle k erfüllt wäre. Nun ergibt sich aber eine völlig neue Lösung der (nun nicht mehr echt quadratischen) Gleichungen 4.125 und 4.127:

$$t_k^{(j)} = \frac{\left(\phi^{-1}(P_{kj})\right)^2}{\ln(c_j P_{kj})} \tag{4.132}$$

$$s_k^{(j)} = \frac{\phi^{-1}(P_{kj})}{\ln(c_j P_{kj})}. \tag{4.133}$$

Für $P_{kj} = 0,5$ ist hier Stetigkeit anzunehmen, siehe dazu die Gleichungen 4.138 und 4.139. Notwendige Lösungsbedingung ist das Vorzeichen des Trainingsfehlers:

$$\begin{aligned} \frac{\phi^{-1}(P_{kj})}{\ln(c_j P_{kj})} &> 0 \\ \phi^{-1}(P_{kj}) > 0 &\iff \ln(c_j P_{kj}) > 0 \\ P_{kj} > 0,5 &\iff c_j P_{kj} > 1 \\ P_{kj} > 0,5 &\iff c_j > P_{kj}^{-1}. \end{aligned} \tag{4.134}$$

Die Wahl

$$c_j = 2 \tag{4.135}$$

erfüllt diese Bedingung für beliebige Werte von P_{kj}. Damit ist durch die Gleichungen 4.132, 4.133 und 4.135 eine Lösung für die gesuchten Konstanten $t^{(j)}$ und $s^{(j)}$ für $j = 1, \ldots, J$ unter Bedingung IIIb gegeben.

Verwendet man Bedingung IIIc ergibt sich aus dieser und Gleichung 4.125 die Lösung

$$t_k^{(j)} = (s^{(j)})^2 \ln(c_j P_{kj}) \tag{4.136}$$

$$s_k^{(j)} = s^{(j)} \tag{4.137}$$

mit noch zu bestimmenden Konstanten c_j und $s^{(j)}$. Um die Lösungen unter den Bedingungen IIIb und IIIc vergleichen zu können, sollen sie für $P_{kj} \approx 0,5$ identisch sein. Mit Hilfe des Satzes von l'Hospital ergibt sich aus den Gleichungen 4.132 und 4.133

$$\begin{aligned} \lim_{P_{kj} \to 0,5} t_k^{(j)} &= \lim_{P_{kj} \to 0,5} \frac{\left(\phi^{-1}(P_{kj})\right)^2}{\ln(2P_{kj})} \\ &= \lim_{P_{kj} \to 0,5} \frac{2\phi^{-1}(P_{kj}) \cdot \frac{\partial(\phi^{-1})}{\partial P_{kj}}(P_{kj})}{P_{kj}^{-1}} \\ &= 0 \end{aligned} \tag{4.138}$$

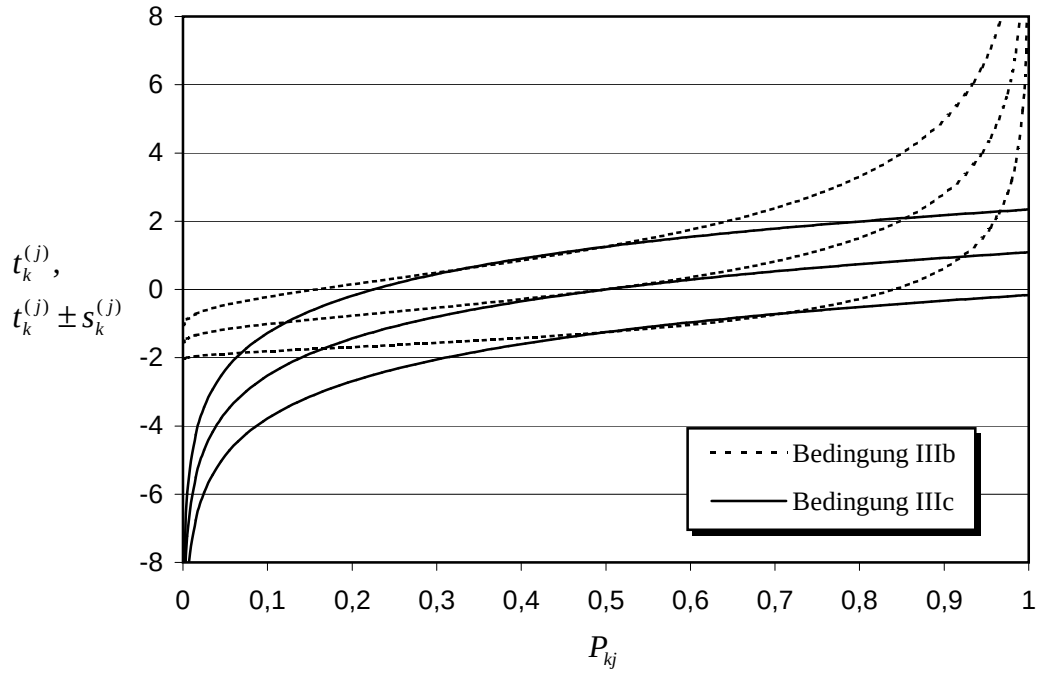

Abbildung 4.6: Abhängigkeit der Trainingswerte und -fehler von der Korrelationswahrscheinlichkeit P_{kj} zwischen beobachteter Klasse D_j und wahrer Klasse C_k als Lösung der Systeme der Bedingungen I/II/IIIb und I/II/IIIc. Dargestellt ist für beide Systeme der jeweilige Trainingswert $t_k^{(j)}$ (mittlere Kurve) zusammen mit dem einfachen Trainingsfehlerintervall $t_k^{(j)} \pm s_k^{(j)}$ (obere und untere Kurve).

$$\begin{aligned}
\lim_{P_{kj}\to 0,5} s_k^{(j)} &= \lim_{P_{kj}\to 0,5} \frac{\phi^{-1}(P_{kj})}{\ln(2P_{kj})} \\
&= \lim_{P_{kj}\to 0,5} \frac{\left(\frac{\partial\phi}{\partial(\phi^{-1}(P_{kj}))}(\phi^{-1}(P_{kj}))\right)^{-1}}{P_{kj}^{-1}} \\
&= \lim_{z\to 0} \frac{1}{2}\left(\frac{\partial\phi}{\partial z}(z)\right)^{-1} \\
&= \lim_{z\to 0} \frac{1}{2}\left(\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}z^2\right)\right)^{-1} \\
&= \sqrt{\frac{\pi}{2}}. \qquad (4.139)
\end{aligned}$$

Hieraus ergeben sich die gesuchten Konstanten zu $c_j = 2$ (wie unter Bedingung IIIb) und $s^{(j)} = \sqrt{\pi/2}$. Der Übersichtlichkeit halber hier die endgültige Lösung unter Bedingung IIIc:

$$t_k^{(j)} = \frac{\pi}{2}\ln(2P_{kj}) \qquad (4.140)$$

$$s_k^{(j)} = \sqrt{\frac{\pi}{2}}. \qquad (4.141)$$

Abbildung 4.6 stellt die Trainingswerte $t_k^{(j)}$ und Trainingsfehler $s_k^{(j)}$ für verschiedene Werte von P_{kj} vergleichend unter den Bedingungen IIIb und IIIc dar.

Gleichung 4.122 definiert nicht nur die Hinabbildung der Trainingsdaten sondern auch die Rückabbildung der Prognosen des Netzes. Die so erhaltenen Prognose-Wahrscheinlichkeiten $h(\mu,\sigma,1),\ldots,h(\mu,\sigma,K)$

für die beobachtbaren Klassen unterliegen — wie Prognosen kontinuierlicher Parameter auch — einem Prognosefehler[6]. Da der Prognosewert μ_k des Netzes mit einem Prognosefehler von σ_k behaftet ist, folgt für die Fortpflanzung $\sigma_k^{(h)}$ dieses Fehlers in Gleichung 4.122 unter Vernachlässigung der Fehler der übrigen Netzausgänge in erster Näherung

$$\begin{aligned}
\sigma_k^{(h)} &:= \sigma_k \frac{\partial}{\partial \mu_k} h(\mu, \sigma, k) \\
&= \sigma_k \frac{\exp(\sigma_k^{-2}\mu_k)\sigma_k^{-2} \cdot \sum_{i=1}^{K} \exp(\sigma_i^{-2}\mu_i) - \exp(\sigma_k^{-2}\mu_k)^2 \sigma_k^{-2}}{\left(\sum_{i=1}^{K} \exp(\sigma_i^{-2}\mu_i)\right)^2} \\
&= \sigma_k^{-1} \exp(\sigma_k^{-2}\mu_k) \frac{\sum_{i=1, i\neq k}^{K} \exp(\sigma_i^{-2}\mu_i)}{\left(\sum_{i=1}^{K} \exp(\sigma_i^{-2}\mu_i)\right)^2} \\
&= \sigma_k^{-1} h(\mu, \sigma, k)(1 - h(\mu, \sigma, k)). \qquad (4.142)
\end{aligned}$$

Zusätzlich ergibt sich ein Fehler durch die Wahrscheinlichkeitsangabe für die Klasse C_k selbst, die je nur entweder die wahre Klasse ist oder nicht. Sei $P := h(\mu, \sigma, k)$ die prognostizierte Wahrscheinlichkeit für C_k. Ist C_k die wahre Klasse, was mit Wahrscheinlichkeit P eintritt, dann würde man sich die Prognose $P = 1$ wünschen; der Fehler der tatsächlichen Prognose ist dann $1 - P$. Ist C_k nicht die wahre Klasse, was mit Wahrscheinlichkeit $1 - P$ eintritt, dann ist der Fehler entsprechend P. Der erwartete mittlere quadratische Fehler $\sigma_k^{(P)}$ ergibt sich dann zu

$$\begin{aligned}
\left(\sigma_k^{(P)}\right)^2 &= P(1-P)^2 + (1-P)P^2 \\
&= P(1-P). \qquad (4.143)
\end{aligned}$$

Der Gesamtfehler ergibt sich aus der euklidischen Summe der beiden unabhängigen Fehlerkomponenten:

$$\begin{aligned}
\sigma_{ges} &= \sqrt{\left(\sigma_k^{(h)}\right)^2 + \left(\sigma_k^{(P)}\right)^2} \\
&= \sqrt{\left(\sigma_k^{-2} h(\mu, \sigma, k)(1 - h(\mu, \sigma, k)) + 1\right) h(\mu, \sigma, k)(1 - h(\mu, \sigma, k))}. \qquad (4.144)
\end{aligned}$$

4.3.3 Empirische Ergebnisse

Die aktuelle Implementierung beinhaltet das in Abschnitt 4.3.2 vorgestellte Verfahren zur Bearbeitung von Klassifikationsproblemen mit mehreren Klassen. Auf eine Implementierung des Zwei-Klassen-Verfahrens nach Abschnitt 4.3.1 wurde verzichtet, da im Korrosionsdatenschema immer mehr als drei beobachtbare Klassen auftraten.

Um das beschriebene und implementierte Verfahren empirisch zu validieren erwiesen sich die vorhandenen Korrosionsdaten als ungeeignet, da keine Vergleichsimplementierung zur Verfügung stand. Daher wurde auf allgemein zugängliche und von anderen Forschergruppen untersuchte Datensammlungen zurückgegriffen. Bei jeder Datenmenge sind Trainings- und Validierungsmenge festgelegt; der Bewertungsmaßstab der verschiedenen Methoden ist der Anteil der falsch klassifizierten Datensätze in der Validierungsmenge. Es wurden die folgenden beiden Datensammlungen verwendet:

Ionosphere (UCI Machine Learning Repository, *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/ionosphere*) 200 Trainingsdatensätze, 151 Validierungsdatensätze, 34 Eingänge, zwei Klassen. Die Klassen namens „good“ und „bad“ entsprechen Radarreflexionen aus der Ionosphäre.
Ob das natürliche Modell für diese Datensammlung eher *class conditional density estimation* (CCDE) oder *discrete-valued function estimation* (DVFE) ist, konnte aus der Beschreibung der Datensammlung nicht klar ermittelt werden.

[6] Es wurde beobachtet, dass seine Darstellung in der Implementierung auf den Anwender eher verwirrend als hilfreich wirkt. Da die Prognosen selbst bereits Wahrscheinlichkeiten darstellen und daher ihre eigene Konfidenz beinhalten, wurde in der Implementierung auf die Darstellung dieser Prognosefehler verzichtet.

$P_=$	99%	97%	95%	90%	80%
mix	23,8	19,2	14,6	9,9	9,3
min	14,6	11,9	9,3	9,3	8,6
max	8,6	8,6	8,6	9,3	9,9
2·max	4,6	6,0	6,0	7,3	9,9
4·max	4,6	6,0	6,0	7,3	9,3
8·max	4,6	4,6	5,3	5,3	6,0
20·max	6,0	5,3	6,0	5,3	6,0
50·max	8,6	6,6	6,0	5,3	5,3

Tabelle 4.2: Ergebnisse für die Ionosphere-Daten. Dargestellt ist der Anteil der falsch klassifizierten Datensätze in der Validierungsmenge in Abhängigkeit der Wahrscheinlichkeit für eine wahre Beobachtung P_{kk} (Spalten) und der Gewichtsregularisierung (Zeilen).

Vowel (Steve Renals' Home Page, *http://www.dcs.shef.ac.uk/∼sjr/com336/assign*) 528 Trainingsdatensätze, 462 Validierungsdatensätze, 10 Eingänge, 11 Klassen. Die Klassen entsprechen 11 englischen Vokalen, die Eingangsvariablen sind aus dem Frequenzspektrum durch lineare Filter extrahierte Features. Die Datensätze wurden durch 15 verschiedene Sprecher erzeugt, von denen jeder jeden Vokal sechsmal sprach. Acht Sprecher bilden dabei die Trainigsmenge, die sieben anderen die Validierungsmenge.
Das natürliche Modell für diese Datensammlung ist *class conditional density estimation* (CCDE), da der Vokal (Netzausgang) dem Sprecher vorgegeben wird und sich die Akustik (Netzeingänge) als Verteilung für den Vokal ergeben.

Bei beiden Datensammlungen wurden Netze nach Abschnitt 3.1 verwendet, deren Gewichtsregularisierung nach Abschnitt 3.2 bestimmt wurde. Der numerische Strafsummand nach Abschnitt 3.2.3 wurde zwar berücksichtigt, war aber in allen Fällen vernachlässigbar klein. Die Menge der Basisfunktionen setzte sich jeweils aus einem Bias (konstant 1), den Eingängen (lineare Basisfunktionen) und einigen zufällig verteilten Hakenfunktionen (Gleichung 3.137) zusammen, sodass die Gesamtzahl der Basisfunktionen jeweils gleich der Anzahl der Trainingsdaten war. Alle Netze hatten identische Basisfunktionen, es wurden Trainingswerte nach Bedingung IIIb (Gleichungen 4.132 und 4.133) verwendet.

Bei den Ionosphere-Daten bestand sowohl die Menge der beobachtbaren als auch die Menge der möglichen wahren Klassen nur aus den Klassen C_{good} und C_{bad}. Um das DVFE Modell anwenden zu können müssen die Korrelationswahrscheinlichkeiten $P_{kj} = P(f = C_k|\varphi_n = C_j)$ mit $k, j \in \{good, bad\}$ festgelegt werden. Diese Wahrscheinlichkeiten sind aber a priori nicht bekannt und konnten auch nicht aus der Beschreibung der Datensammlung ermittelt werden. Eine sinnvolle Festlegung ist die Symmetrie der Korrelationswahrscheinlichkeiten P_{kj}: sei $P_=$ die Wahrscheinlichkeit, dass die beobachtete Klasse auch die wahre Klasse ist, dann wurde

$$P_{good,good} = P_{bad,bad} = P_= \tag{4.145}$$
$$P_{good,bad} = P_{bad,good} = 1 - P_= \tag{4.146}$$

festgelegt. Da auch $P_=$ nicht a priori geschätzt werden konnte, wurden verschiedene Werte getestet.

Das verwendete DVFE Modell prognostiziert für jede der beiden Klassen C_{good} und C_{bad} eine a posteriori Wahrscheinlichkeit. Aus Gründen der Vergleichbarkeit wurde als prognostizierte Klasse diejenige mit der größeren a posteriori Wahrscheinlichkeit verwendet.

Tabelle 4.2 listet die Rate der Falschklassifikation bei der Prognose in der Validierungsmenge, hier kurz Fehlerrate genannt, auf. Zunächst wurden für beide Netzausgänge die optimalen Gewichtsregularisierungen ermittelt; in allen Spalten war $(\sigma_\mathrm{w})_{good} < (\sigma_\mathrm{w})_{bad}$. In der mit mix bezeichneten Zeile wurden beide Ausgänge individuell regularisiert, sie besaßen also die Gewichtsregularisierungen $(\sigma_\mathrm{w})_{good}$ bzw. $(\sigma_\mathrm{w})_{bad}$. Hier war nicht nur die Fehlerrate auf der Validierungsmenge, sondern sogar die Fehlerrate auf der Trainingsdatenmenge hoch. In allen weiteren Zeilen besaßen beide kontinuierlichen Ausgänge die gleiche Gewichtsregularisierung: in Zeile min die kleinere, $(\sigma_\mathrm{w})_{good}$, und in Zeile $(\sigma_\mathrm{w})_{bad}$ die größere. Da ein

$P_=$	97%	90%
P_{kj} für $k \neq j$	0,3%	1%
mix	70,1	69,9
avg	58,7	59,7
2·avg	53,7	54,1
4·avg	51,3	51,1
8·avg	49,8	49,4
16·avg	48,1	47,8
32·avg	45,7	46,8
64·avg	44,6	45,0
128·avg	46,3	46,3
256·avg	47,2	47,6

Tabelle 4.3: Ergebnisse für die Vowel-Daten

größerer Wert für σ_w offensichtlich zu kleineren Fehlerrate führte, wurden weitere größere Werte, nämlich Vielfache von $(\sigma_\mathrm{w})_{bad}$ getestet.

Die Fehlerraten auf der Validierungsmenge aus Tabelle 4.2 können mit den in [PenRob] aufgeführten Fehlerraten verglichen werden. Dort wird eine Fehlerrate von 7,3% bei Verwendung eines einzelnen Netzes (mit mehreren Ausgängen) und verschiedener Komitees berichtet. Lediglich bei Verwendung eines speziellen Komitees (3,3%) oder unter Verwendung von *automatic relevance determination* (4,0%) konnte diese Fehlerrate verringert werden.

Die Vowel-Datensammlung wurde ähnlich untersucht. Erwähnenswert ist hier lediglich die Wahl der Korrelationswahrscheinlichkeiten P_{kj} mit $k, j \in 1, \ldots, 11$. Die Wahl

$$P_{kj} = \begin{cases} P_= & \text{, falls } k = j \\ \frac{1}{10}(1 - P_=) & \text{, falls } k \neq j \end{cases} \tag{4.147}$$

mit verschiedenen Werten für $P_=$ ist wieder symmetrisch in allen Klassen.

Tabelle 4.3 zeigt die Fehlerraten für die Vowel-Daten. In Zeile mix ist jeder der 11 kontinuierlichen Ausgänge individuell regularisiert, avg bezeichnet eine gemeinsame Gewichtsregularisierung in Höhe des arithmetischen Mittels der individuellen Gewichtsregularisierungen. Als Vergleich dazu berichtet [PenRob] von Fehlerraten von 70,1% bei Einzelnetzen und 46,1% bis 50,9% bei Komitees.

Zusammengefasst können diese empirischen Untersuchungen wie folgt bewertet werden:

- Vergleicht man DVFE mit CCDE aus [PenRob], so können mit DVFE sehr niedrige Fehlerraten auf der Validierungsmenge erreicht werden. Dies ist erstaunlich, da DVFE zumindest bei den Vowel-Daten eigentlich kein adäquates Modell ist. Für die Probleme der Korrosion ist daher eher ein besseres Verhalten anzunehmen.

- Zwar können niedrige Fehlerraten erreicht werden, jedoch konnte eine automatische Einstellung der Parameter $P_=$ und σ_w hier nicht gefunden worden. Dies steht leider im Gegensatz zu Netzen, die Regressionsprobleme lösen, und die vollautomatisch, also ohne manuelle Einstellung von Netzparametern, trainiert werden können. Es besteht daher an dieser Stelle weiterer Forschungsbedarf.

- Die Fehlerrate ist offensichtlich gering von der Korrelationswahrscheinlichkeit $P_=$ und stark von der Gewichtsregularisierung σ_w abhängig. Während $P_=$ bzw. allgemeiner P_{kj} eigentlich modellabhängige Konstanten sind, also a priori Wissen über das Problem ausdrücken, sollte σ_w automatisch einstellbar sein. Es gibt folgende mögliche Erklärungsansätze, warum das optimale σ_w für das Klassifikationsproblem signifikant größer sein sollte als das automatisch vom Netz unter Annahme eines Regressionsproblems gefundene:

  1. Es ist bekannt ([ImpSal], [PerCoo]), dass sich Overfitting, also „zu großes“ σ_w, bei Komitees von Netzen positiv auf die Gesamtprognose auswirken kann. Bei K möglichen wahren Klassen bilden die K kontinuierlichen Ausgänge ein Ensemble von Netzen. Die Wirkungsweise dieses

Ensembles ist zwar anders als die eines Komitees, möglicherweise stellt sich aber hier ein ähnlicher Effekt ein.

2. Das Verhalten der Netze wurde durch die Gleichungen 4.112 und 4.113 beschrieben, die aber nur ein vereinfachtes Modell des Netzes beschreiben. Es wurde gezeigt, dass diese Gleichungen die Netze aus Abschnitt 3.1 umso besser beschreiben, je größer σ_w ist, also je vernachlässigbarer die Gewichtsregularisierung ist.
3. Bedingung IIIb stellt eine Näherung von Bedingung IIIa dar. Es wäre zu prüfen, ob ein Wechsel zu Bedingung IIIa Einfluss auf die Fehlerraten hat.

- Bei den Vowel-Daten könnte eine Verbesserung erreicht werden, wenn man die Korrelationswahrscheinlichkeiten P_{kj} problemangepasster wählen würde. Einige gesprochene Vokale sind sich ähnlicher als andere: so verwechselt man die Wörter „put“ und „pot“ leichter als die Wörter „put“ und „pit“. Es gibt daher eine nicht-symmetrische Ähnlichkeitsrelation der 11 Vokale, anhand derer man die Korrelationswahrscheinlichkeiten wählen könnte. Man beachte, dass es sich bei dieser Ähnlichkeitsrelation um a priori Wissen über das Problem handelt. Konkret ist dieses Wissen aber nicht vorhanden, nicht einmal die Zuordnung der Klassenmarkierung und des zugehörigen Vokals war der Beschreibung der Vowel-Datensammlung zu entnehmen.

In [KonDie] und [UtsWei] werden redundante Kodierungen der beobachtbaren Klassen mit klassischen Netzen untersucht, es konnten verbesserte Generalisierungseigenschaften erzielt werden. Es besteht daher die Hoffnung, dass sich durch die Übertragung dieser Kodierungstechniken auf das hier vorgestellte DVFE-Modell weitere Verbesserungen ergeben könnten.

Hier, am Ende des Abschnitts 4.3, sollen noch einmal kurz die wichtigsten Unterschiede zwischen CCPE und DVFE zusammengefasst werden.

- CCPE modelliert zufällige Eingangsvariablen, deren Verteilung von der Klasse als Ausgangsgröße abhängt. Jeder Eingangsvektor kann mit mehreren Klassen assoziiert sein. DVFE nimmt eine deterministische Abbildung der Eingänge (Messstelle) auf genau eine Klasse an.
- Bei CCPE assoziiert jeder Trainingsdatensatz einen Eingangsdatenvektor mit genau einer Klasse. Bei DVFE wird mit jedem Trainingsdatensatz jeder Messstelle eine Verteilung der Ausgangsklassen zugeordnet. Daher enthält bei DVFE jeder einzelne Trainingsdatensatz wesentlich mehr Information als bei CCPE.
- Wird DVFE zusammen mit generalisierten linearen Netzen angewendet, ist das Training für ein gegebenes σ_w sehr schnell, da alle Berechnungen analytisch, also nicht-iterativ durchgeführt werden können. CCPE führt dagegen — soweit bekannt — immer zu einer analytisch nicht lösbaren Fehlerfunktion.

4.4 Regionales Rauschen

[7]Neuronale Netze mit bayesschen Methoden können für kontinuierliche Ausgangsgrößen neben Prognosewerten $\mu(x)$ auch Prognosevarianzen $\sigma^2(x)$ berechnen. Dabei wird angenommen, dass es eine wahre Funktion f gibt, die jeder Stelle den eindeutig bestimmten Wert der Ausgangsgröße zuordnet. Nur unter dieser und weiteren Annahmen gilt

$$\mu(x) \approx \mathcal{N}\big(f(x), \sigma^2(x)\big). \tag{4.148}$$

Man beachte, dass dieser Ausdruck eine wesentliche praxisrelevante Forderung an ein Netz beschreibt: das Netz soll Prognosewerte berechnen, die in der Nähe der wahren Werte liegen und es soll den Abstand zwischen beiden einschätzen können.

Nun gibt es im Falle der Korrosion eben keine derartige wahre Funktion f. Dies hat folgende mögliche Ursachen:

[7]Wesentliche Teile dieses Abschnitts wurden bereits in [WebSchSch] vorveröffentlicht.

- Das Korrosionsverhalten unterliegt inhärentem Rauschen, siehe dazu [CotQinOwe]. Ursachen dafür sind der stochastische Charakter chemischer Korrosionsreaktionen und (in der Praxis) nicht bestimmbare Einflussgrößen der Korrosion, wie beispielsweise die Güte der Verteilung der Legierungselemente im Werkstück oder geheimzuhaltende Zusätze im Medium.

- Einige Korrosionssysteme können bistabile elektrochemische Zustände bilden: je nach den Anfangsbedingungen können die Systeme dauerhaft den aktiven oder den passiven Zustand annehmen. Siehe dazu [DIN50900] oder [Gräfen].

Es muss hier betont werden, dass die Korrosion überwiegend, d.h. an den meisten Stellen, in hinreichend guter Näherung Funktionscharakter hat. Die wenigen besonderen Stellen dagegen sind nicht a priori bekannt. Somit können die beim Training benötigten Messfehler nur aufgrund der technischen Eigenschaften der Messapparatur bestimmt werden und spiegeln daher nicht das Rauschen des Korrosionsverhaltens wieder.

Die Abweichung des Korrosionsverhaltens vom Modell hat weitreichende Konsequenzen auf die Interpretation der Prognosen der Netze aus Abschnitt 3.1. Setzt man voraus, dass eine adäquate Menge von Messungen durchgeführt wurde, die die wahre Verteilung gut beschreibt — also etwa gleich viele aktivierte und passivierte Systeme, falls Bistabilität möglich ist —, dann beschreibt der Prognosewert $\mu(x)$ lediglich eine a posteriori Schätzung des mittleren Korrosionsverhaltens. Dies ist in der Praxis zwar nicht wirklich tragisch, viel problematischer aber ist, dass die Prognosevarianz $\sigma^2(x)$ erheblich zu klein ausfällt. Wie aus Abschnitt 3.1 bekannt, hängt $\sigma^2(x)$ allein von den Messstellen und Messfehlern, nicht aber von den Messwerten ab, was dazu führt, dass die Prognosefehler unabhängig davon sind, ob die Messwerte untereinander widersprüchlich[8] bezüglich ihrer Messfehler sind oder nicht. Lemma 2 in Anhang B zeigt, dass bei beliebiger Gewichtsregularisierung σ_{w} der Prognosefehler $\sigma(x_n)$ an einer Messstelle x_n immer kleiner als der Messfehler s_n ist.

Im Folgenden soll ein Modell für das inhärente Rauschen entwickelt werden. Dieses Modell ersetzt die wahre Funktion $f(x)$ durch eine ortsabhängige Normalverteilung mit Erwartungswert $f(x)$ und Varianz $\phi^2(x)$. Nach diesem Modell ist die Verteilung eines Messwerts t_n an der Stelle x_n mit Messfehler s_n durch

$$t_n \quad \propto \quad \mathcal{N}\left(f(x_n), \phi^2(x_n) + s_n^2\right) \tag{4.149}$$

gegeben. Natürlich sind sowohl f als auch ϕ unbekannt und zu ermitteln.

Auch über die Verteilung der Trainingsstellen müssen nun explizite Modellannahmen getroffen werden: es wird angenommen, dass $\phi(x) > 0$ nur an Stellen x ist, die in der Nähe von mehreren, sich widersprechenden Trainingsdaten liegen. Diese Annahme ist notwendig, da die Menge der zur Verfügung stehenden Trainingsdaten natürlich knapp ist und Messungen nur dort vorgenommen wurden, wo noch kein Wissen in Form von anderen Messungen vorhanden war. Andererseits ist diese Annahme aber durch die bewusste Auswahl der Messstellen durch die Korrosionsingenieure gerechtfertigt, die nämlich aus Erfahrung und theoretischen Überlegungen heraus mögliche verrauschte oder bistabile Korrosionssysteme vorhersagen können und dann dort gezielt Mehrfachmessungen durchführen.

Während der Prognosewert auch unter Annahme inhärenten Rauschens der gleiche wie der der Netze nach Abschnitt 3.1 sein kann, wird zusammenfassend folgendes Verhalten vom Prognosefehler, hier im Sinne einer Prognosekonfidenz, gefordert:

1. Abseits der Trainingsdaten soll der Prognosefehler groß sein.

2. In der Nähe eines einzelnen, isolierten Trainingsdatensatzes soll der Prognosefehler so groß wie der Trainingsfehler sein.

3. In der Nähe vieler, sich nicht widersprechenden Trainingsdaten soll der Prognosefehler klein sein.

4. In der Nähe vieler widersprüchlicher Trainingsdaten soll der Prognosefehler etwa so groß wie die Standardabweichung der Trainingswerte sein.

[8] Der Begriff widersprüchlich wird im Folgenden verwendet, um Trainingsdaten zu beschreiben, deren Stellen nahe beieinander liegen und deren Abweichung der Trainingswerte voneinander deutlich größer ist als ihre Messfehler. Es handelt sich also um genau solche Messdaten, die nur durch ein Rauschen in der wahren „Funktion“ erklärt werden können.

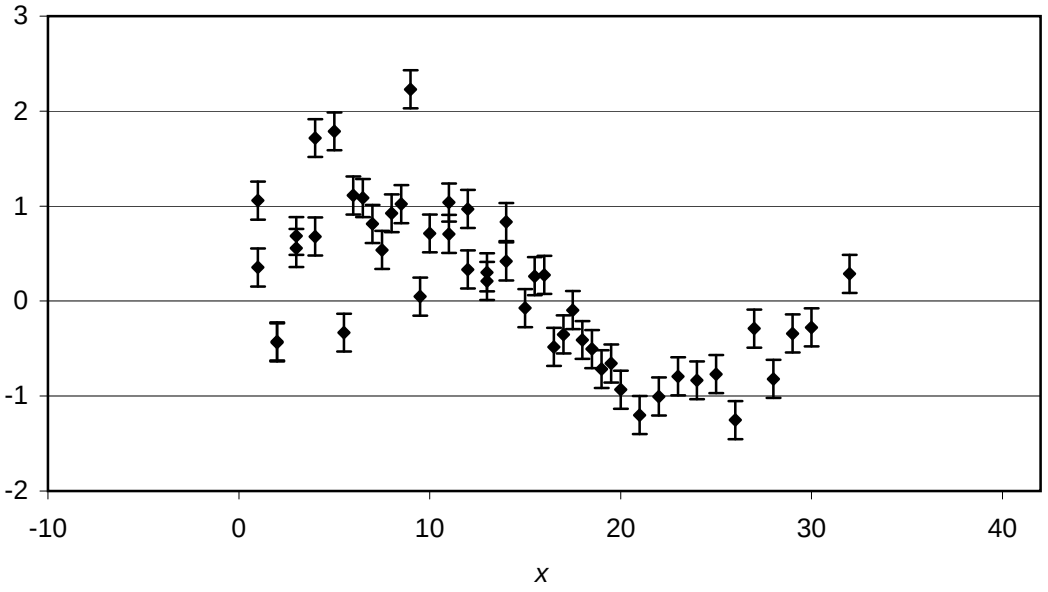

Abbildung 4.7: Beispiel für Daten mit regionalem Rauschen. Offensichlich schwanken die Messwerte im linken Bereich erheblich stärker als ihre Messfehler zulassen würden.

Die Netze nach Abschnitt 3.1 erfüllen bereits die Anforderungen 1–3. Um auch die letzte Forderung zu erfüllen, muss ein neues Modell geschaffen und algorithmisch umgesetzt werden.

In der Literatur wurden bereits verschiedene Verfahren diskutiert, um regional unterschiedliches Rauschen in den Trainingsdaten zu erkennen und in die Prognose aufzunehmen: [CarCunBha], [DybRob], [FoxCawTal], [Heskes], [NixWei] und [WeiNix]. Diese erfüllen aber nicht alle oben genannten Forderungen, in der Regel die ersten beiden nicht. Zudem benötigen sie sehr viele, dichte Trainingsdaten, während die vorhandenen Korrosionsdaten der KISS-Datenbank den Eingangsraum sehr spärlich bevölkern.

Die folgenden Unterabschnitte beschreiben verschiedene Lösungsansätze. Allen gemein ist, dass sie — wie bei den diskontinuierlichen Ausgangsgrößen in Abschnitt 4.3 — die Probleme punktweise betrachten. Es wird nur eine einzige Stelle betrachtet und das Problem dort gelöst, die Verallgemeinerung auf eine Stellenabhängigkeit geschieht dann anschließend durch die Netze. Daher wird im Folgenden die Notation der Stelle x der Einfachheit halber weggelassen, Ausdruck 4.149 lautet dann

$$t_n \quad \propto \quad \mathcal{N}\left(f, \phi^2 + s_n^2\right). \tag{4.150}$$

4.4.1 Ein allgemeines Modell

Ausdruck 4.150 beschreibt direkt die Verteilung der Messwerte bei bekannter wahrer Funktion f und wahrer regionaler Rauschvarianz ϕ^2. Es liegt nun nahe, die bayesschen Methoden hier direkt anzuwenden und aus dieser Verteilung eine a posteriori Verteilung von Gewichten herzuleiten.

Dazu soll f durch die Netzfunktion $g(w)$ und ϕ^2 durch die Rauschfunktion $\psi(w)$ geschätzt werden. Beide Funktionen teilen sich einen gemeinsamen Gewichtsvektor w, was jedoch nicht ausschließt, dass jede Komponente von w in nur je eine der beiden Funktionen g und ψ eingeht. Die Verteilung eines Messwerts bei bekanntem wahren Gewichtsvektor w ergibt sich nun zu

$$p(t_n|w) \quad = \quad \frac{1}{\sqrt{2\pi(\psi(w) + s_n^2)}} \exp\left(-\frac{(t_n - g(w))^2}{2(\psi(w) + s_n^2)}\right). \tag{4.151}$$

Über die bayessche Gleichung erhält man die a posteriori Gewichtsverteilung

$$p(w|D) \quad = \quad \frac{p(w)}{p(D)} \prod_{n=1}^{N} p(t_n|w)$$

$$= \frac{p(w)}{p(D)} \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi(\psi(w)+s_n^2)}} \exp\left(-\frac{(t_n-g(w))^2}{2(\psi(w)+s_n^2)}\right) \tag{4.152}$$

sowie die zugehörige Fehlerfunktion

$$\begin{aligned} S(w) &= -\ln p(w|D) + const \\ &= -\ln p(w) + \frac{1}{2}\sum_{n=1}^{N} \ln(\psi(w)+s_n^2) + \frac{1}{2}\sum_{n=1}^{N} \frac{(t_n-g(w))^2}{\psi(w)+s_n^2}. \end{aligned} \tag{4.153}$$

Einige Eigenschaften dieses Modells sollen nun an einem einfachen Beispiel erklärt werden. Dazu sei $s_1 = \ldots = s_N =: s$, $w \in \mathbb{R}^2$, $p(w) = const$, $g(w) = w_1$ und $\psi(w) = w_2$. Für die Trainingsdaten seien der Mittelwert und die mittlere quadratische Abweichung durch die Größen

$$\bar{t} := \frac{1}{N}\sum_{n=1}^{N} t_n \tag{4.154}$$

$$\delta^2 := \frac{1}{N}\sum_{n=1}^{N} (t_n-\bar{t})^2 \tag{4.155}$$

bezeichnet. Die Fehlerfunktion lautet nun

$$\begin{aligned} S(w) &= \frac{1}{2}\sum_{n=1}^{N} \ln(w_2+s^2) + \frac{1}{2}\sum_{n=1}^{N} \frac{(t_n-w_1)^2}{w_2+s^2} \\ &= \frac{N}{2}\ln(w_2+s^2) + \frac{1}{2(w_2+s^2)}\sum_{n=1}^{N}\left((t_n-\bar{t})^2 + (t_n-\bar{t})(\bar{t}-w_1) + (\bar{t}-w_1)^2\right) \\ &= \frac{N}{2}\ln(w_2+s^2) + \frac{1}{2(w_2+s^2)}\left(N\delta^2 + N(\bar{t}-w_1)^2\right). \end{aligned} \tag{4.156}$$

Um $S(w)$ an der Stelle ihres Minimums quadratisch zu approximieren, werden die folgenden Ableitungen benötigt:

$$\nabla S(w) = \begin{pmatrix} \dfrac{N(w_1-\bar{t})}{w_2+s^2} \\ \dfrac{N}{2(w_2+s^2)} - \dfrac{N\delta^2+N(\bar{t}-w_1)^2}{2(w_2+s^2)^2} \end{pmatrix} \tag{4.157}$$

$$\nabla\nabla S(w) = \begin{pmatrix} \dfrac{N}{w_2+s^2} & -\dfrac{N(w_1-\bar{t})}{(w_2+s^2)^2} \\ -\dfrac{N(w_1-\bar{t})}{(w_2+s^2)^2} & -\dfrac{N}{2(w_2+s^2)^2} + \dfrac{N\delta^2+N(\bar{t}-w_1)^2}{(w_2+s^2)^3} \end{pmatrix}. \tag{4.158}$$

Daraus ergibt sich in dieser Approximation

$$w_{\mathrm{MP}} = \begin{pmatrix} \bar{t} \\ \delta^2 - s^2 \end{pmatrix} \tag{4.159}$$

$$\nabla\nabla S(w_{\mathrm{MP}}) = \begin{pmatrix} \dfrac{N}{\delta^2} & 0 \\ 0 & \dfrac{N}{2\delta^4} \end{pmatrix} \tag{4.160}$$

$$S(w) \approx const + \frac{N}{2\delta^2}(w_1-\bar{t})^2 + \frac{N}{4\delta^4}(w_2+s^2-\delta^2)^2 \tag{4.161}$$

$$p(w_1|D) \propto \mathcal{N}\left(\bar{t}, \frac{\delta^2}{N}\right) \tag{4.162}$$

$$p(w_2|D) \propto \mathcal{N}\left(\delta^2-s^2, \frac{2\delta^4}{N}\right) \tag{4.163}$$

und $w_1|D$ und $w_2|D$ sind stochastisch unabhängig. Diese Verteilungen beschreiben gleichzeitig auch die Prognosen, da $g(w) = w_1$ und $\psi(w) = w_2$ ist. Der Erwartungswert für den wahren mittleren Wert f wird mit $\bar{t}$ gut in Form eines erwartungstreuen Schätzers angegeben. Für viele Datensätze ($N \to \infty$) ist auch das Verhalten der Varianzen intuitiv: je mehr Daten, desto genauer können der mittlere Wert f und die Varianz des Rauschens ϕ^2 eingeschätzt werden.

Es gibt allerdings auch einige Aspekte im Prognoseverhalten, die nicht erwünscht sind:

- Der prognostizierte Erwartungswert für die Varianz des regionalen Rauschens ist $\delta^2 - s^2$ und kann durchaus negativ werden, da δ nur von den Messwerten, nicht aber vom Messfehler s abhängt. Dies könnte durch Modifikation der Funktion ψ behoben werden: z.B. $\psi(w) = \exp(\psi_0(w))$. Dann ist aber die Existenz eines Minimums von $S(w)$ nur noch unter echter Gewichtsregularisierung garantiert.

- Der Erwartungswert für die Rauschvarianz ϕ^2 ist mit $\delta^2 - s^2$ nicht erwartungstreu. Man beachte, dass unter Normalverteilungsannahme bei unbekanntem Erwartungswert die Größe

$$\tilde{\delta}^2 \quad := \quad \frac{1}{N-1} \sum_{n=1}^{N} (t_n - \bar{t})^2 \tag{4.164}$$

  ein erwartungstreuer Schätzer der Gesamtvarianz (Varianz der $t_1, \ldots, t_N$) ist. Nun ist zwar $\delta \approx \tilde{\delta}$ für $N \to \infty$, allerdings sind gerade kleine N interessant, da in der Regel nur wenige Messstellen in Nähe zueinander liegen.

- Forderung 2 auf Seite 99 korrespondiert mit dem Fall $N = 1$, also einer isolierten, von allen anderen Messstellen weit entfernten Messstelle. Erwartet würde hier, dass eine Bestimmung des regionalen Rauschens prinzipiell nicht möglich ist. Das Gegenteil ist aber der Fall: für $N = 1$ ist $\bar{t} = t_1$ und $\delta = 0$. Es folgt nicht nur eine geschätzte Varianz des Rauschens von $-s^2$, insbesondere verschwinden auch die Prognosefehler für g und ψ.

Bei einer realen Implementierung müsste ein Minimum der Funktion $S(w)$, Gleichung 4.153, algorithmisch berechnet werden. Dazu ist derzeit kein analytisches Verfahren bekannt, vielmehr muss auf ein iteratives Verfahren zur Minimumsuche, etwa konjugierte Gradienten, zurückgegriffen werden. Dies hat allerdings zwei schwerwiegende Nachteile: einerseits wird die Laufzeit durch die Iterationslösung deutlich vergrößert und andererseits muss stets sichergestellt werden, dass das gefundene lokale Minimum akzeptabel, d.h. idealerweise das globale Minimum, ist. Diese Probleme und die aus dem Beispiel abgeleiteten ungünstigen Eigenschaften bei der Prognose haben dazu geführt, dieses Modell nicht zu implementieren und einen anderen Weg zu suchen.

4.4.2 Schätzer für identische Messfehler

Wie bereits zu Anfang des Abschnitts 4.4 aufgeführt, leistet bereits ein einzelnes Netz nach Abschnitt 3.1 gute Prognosen, wenn sich die Trainingsdaten nicht widersprechen. Es liegt daher nahe, ein solches Netz direkt zu verwenden und es lediglich durch ein zweites Netz zu ergänzen, das die Varianz des regionalen Rauschens prognostizieren soll. Bevor der allgemeine Fall diskutiert wird, der eine Näherung erfordert, soll hier ein spezieller Fall, der ohne Näherung lösbar ist, vorgestellt werden.

Seien $t_1, \ldots, t_N$ Messwerte, die alle mit dem Messfehler s gemessen wurden. Für die zu bestimmenden Größen f und ϕ^2 wird nun je ein Netz verwendet. Das Netz namens f wird mit den Messwerten $t_1, \ldots, t_N$ und den Messfehlern $s, \ldots, s$ trainiert, man erhält von ihm folgende Prognosen:

$$\mu_f \quad = \quad \frac{1}{N} \sum_{n=1}^{N} t_n \tag{4.165}$$

$$\sigma_f^2 \quad = \quad \frac{s^2}{N}. \tag{4.166}$$

Diese Gleichungen lassen sich leicht aus den Näherungsgleichungen für Netze 3.100 und 3.101 ableiten. Sie beschreiben hier alle Annahmen, die über das Verhalten der Netze getroffen werden.

Aus der Verteilung der Messwerte nach 4.150 kann nun die Verteilung des Prognosewerts bestimmt werden, sie ist

$$\mu_f \quad \propto \quad \mathcal{N}\left(f, \frac{\phi^2}{N} + \sigma_f^2\right). \tag{4.167}$$

Nachdem durch das Netz f bereits der mittlere wahre Wert f erwartungstreu geschätzt wird, bleibt als Ziel nun die praktische Bestimmung der Varianz des regionalen Rauschens ϕ^2 im Rahmen einer Prognose. Um dieses Ziel zu erreichen, werden nach dem Training des Netzes f Prognosen an allen Messstellen zu allen Trainingsdaten berechnet. Aus diesen Prognosen μ_f und σ_f^2 und den ursprünglichen Trainingsdaten werden nun die Größen

$$u_n \quad := \quad \frac{s^2}{s^2 - \sigma_f^2}(t_n - \mu_f)^2 - s^2 \quad \text{für} \quad n = 1, \ldots, N. \tag{4.168}$$

berechnet. Man beachte, dass für $N = 1$ die Größe u_1 nicht definiert ist, da im Minuend sowohl der Nenner als auch der zweite Faktor verschwinden. Regionales Rauschen kann selbstverständlich mit nur einer Messung nicht bestimmt werden.

Der Erwartungswert von u_n bezüglich der Messwerte ergibt sich für jedes $n = 1, \ldots, N$ zu

$$\begin{aligned}
E[u_n|f] &= \frac{s^2}{s^2 - \sigma_f^2} E\left[(t_n - \mu_f)^2 | f\right] - s^2 \\
&= \frac{s^2}{s^2 - s^2/N} E\left[\left(t_n - \frac{1}{N}\sum_{i=1}^{N} t_i\right)^2 \mid f\right] - s^2 \\
&= \frac{N}{N-1} E\left[\left(t_n - f - \frac{1}{N}\sum_{i=1}^{N}(t_i - f)\right)^2 \mid f\right] - s^2 \\
&= \frac{N}{N-1} E\left[(t_n - f)^2 - 2(t_n - f)\frac{1}{N}\sum_{i=1}^{N}(t_i - f) + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(t_i - f)(t_j - f) \mid f\right] - s^2 \\
&= \frac{N}{N-1} E\left[(t_n - f)^2 - 2\frac{1}{N}(t_n - f)^2 + \frac{1}{N^2}\sum_{i=1}^{N}(t_i - f)^2 \mid f\right] - s^2 \\
&= \frac{N}{N-1}\left(1 - 2\frac{1}{N} + \frac{N}{N^2}\right)(\phi^2 + s^2) - s^2 \\
&= \frac{N}{N-1}\frac{N-1}{N}(\phi^2 + s^2) - s^2 \\
&= \phi^2.
\end{aligned} \tag{4.169}$$

Trainiert man nun das Netz namens ϕ mit den „Messwerten“ $u_1, \ldots, u_N$ und identischen „Messfehlern“ $v, \ldots, v \in \mathbb{R}^+$, dann ist sein Prognosewert ein erwartungstreuer Schätzer für die Varianz des regionalen Rauschens ϕ^2.

4.4.3 Schätzer für unterschiedliche Messfehler

Die in Abschnitt 4.4.2 vorgestellte Behandlung des regionalen Rauschens ist nicht auf stellenabhängige Probleme übertragbar. Dies liegt an der gegenseitigen Beeinflussung verschiedener Messstellen untereinander: auf eine Prognose an der Stelle x wirkt eine Messung an einer benachbarten Stelle $x_n \approx x$ ähnlich einer Messung an gleicher Stelle x, jedoch mit vergrößertem Messfehler (Abschnitt 3.3.7). Die genaue Wirkung ist kompliziert, man wird jedoch erwarten, dass diese ortsbedingte fiktive Vergrößerung des Messfehlers mit dem Abstand der Stellen x und x_n zunimmt.

Seien nun $t_1, \ldots, t_N$ Messwerte, die unter den expliziten Messfehlern $s_1, \ldots, s_N$ gemessen wurden. Das Netz f wird nun wie gehabt mit den Messwerten $t_1, \ldots, t_N$ und den Messfehlern $s_1, \ldots, s_N$ trainiert, es

berechnet gemäß den Näherungsgleichungen für Netze 3.100 und 3.101 dann die Prognose

$$\mu_f = \sigma_f^2 \sum_{n=1}^{N} s_n^{-2} t_n \tag{4.170}$$

$$\sigma_f = \left(\sum_{n=1}^{N} s_n^{-2} \right)^{-1}. \tag{4.171}$$

Der Prognosewert μ_f spielt im Folgenden eine wichtige Rolle, da u.a. aus ihm die Trainingswerte des zweiten Netzes ϕ berechnet werden sollen. Offensichtlich ist sein Erwartungswert über viele Messreihen gerade der mittlere wahre Wert, $E[\mu_f|f] = f$, da er eine Konvexkombination über die Messwerte $t_1, \ldots, t_N$ mit gleichem Erwartungswert darstellt. Die mittlere Abweichung zwischen dem Prognosewert und einem Messwert kann wie folgt berechnet werden:

$$\begin{aligned}
E\left[(t_n - \mu_f)^2|f\right] &= E\left[\left(t_n - \sigma_f^2 \sum_{i=1}^{N} s_i^{-2} t_i \right)^2 \,\middle|\, f \right] \\
&= E\left[\left(t_n - f - \sigma_f^2 \sum_{i=1}^{N} s_i^{-2} (t_i - f) \right)^2 \,\middle|\, f \right] \\
&= E\left[\left((1 - \sigma_f^2 s_n^{-2})(t_n - f) - \sigma_f^2 \sum_{i \neq n} s_i^{-2} (t_i - f) \right)^2 \,\middle|\, f \right] \\
&= (1 - \sigma_f^2 s_n^{-2})^2 E\left[(t_n - f)^2|f\right] + \sum_{i \neq n} \sigma_f^4 s_i^{-4} E\left[(t_i - f)^2|f\right] \\
&= (1 - \sigma_f^2 s_n^{-2})^2 (\phi^2 + s_n^2) + \sum_{i \neq n} \sigma_f^4 s_i^{-4} (\phi^2 + s_i^2) \\
&= \left((1 - \sigma_f^2 s_n^{-2})^2 + \sum_{i \neq n} \sigma_f^4 s_i^{-4} \right) \phi^2 + (1 - \sigma_f^2 s_n^{-2})^2 s_n^2 + \sum_{i \neq n} \sigma_f^4 s_i^{-4} s_i^2 \\
&= \left(1 - 2\sigma_f^2 s_n^{-2} + \sigma_f^4 s_n^{-4} + \sum_{i \neq n} \sigma_f^4 s_i^{-4} \right) \phi^2 + s_n^2 - 2\sigma_f^2 + \sigma_f^4 s_n^{-2} + \sigma_f^4 (\sigma_f^{-2} - s_n^{-2}) \\
&= \left(1 - 2\sigma_f^2 s_n^{-2} + \sigma_f^4 \sum_{i=1}^{N} s_i^{-4} \right) \phi^2 + s_n^2 - \sigma_f^2.
\end{aligned} \tag{4.172}$$

Dieser Ausdruck lässt sich nicht weiter vereinfachen, da die Summe über i aufgrund des Exponenten -4 der Messfehler nicht berechnet werden kann. Daher wird eine Näherung verwendet, die die Summanden aufspaltet: $s_i^{-4} \approx s_i^{-2} s_n^{-2}$. Es ist offensichtlich, dass die Näherung exakt ist, falls alle Messfehler (bzw. ihre Wirkung auf die Prognosestelle) gleich sind. Die allgemeine Genauigkeit der Näherung wird weiter unten noch diskutiert.

$$\begin{aligned}
E\left[(t_n - \mu_f)^2|f\right] &\approx \left(1 - 2\sigma_f^2 s_n^{-2} + \sigma_f^4 \sum_{i=1}^{N} s_i^{-2} s_n^{-2} \right) \phi^2 + s_n^2 - \sigma_f^2 \\
&= \left(1 - 2\sigma_f^2 s_n^{-2} + \sigma_f^4 \sigma_f^{-2} s_n^{-2} \right) \phi^2 + s_n^2 - \sigma_f^2 \\
&= (1 - \sigma_f^2 s_n^{-2}) \phi^2 + s_n^2 - \sigma_f^2 \\
&= (1 - \sigma_f^2 s_n^{-2})(\phi^2 + s_n^2)
\end{aligned} \tag{4.173}$$

Seien nun wie im vorigen Abschnitt die Trainingswerte des Netzes ϕ durch

$$u_n := \frac{(t_n - \mu_f)^2}{1 - \sigma_f^2 s_n^{-2}} - s_n^2 \qquad \text{für} \qquad n = 1, \ldots, N \tag{4.174}$$

gegeben. Der Trainingswert u_n besitzt über viele Messreihen den exakten Erwartungswert nach Gleichung 4.172

$$\begin{aligned} E\left[u_n|f\right] &= \frac{\left(1-2\sigma_f^2 s_n^{-2}+\sigma_f^4\sum_{i=1}^N s_i^{-4}\right)\phi^2+s_n^2-\sigma_f^2}{1-\sigma_f^2 s_n^{-2}} - s_n^2 \\ &= \left(\frac{1-2\sigma_f^2 s_n^{-2}}{1-\sigma_f^2 s_n^{-2}} + \sum_{i=1}^N \frac{\sigma_f^4 s_i^{-4}}{1-\sigma_f^2 s_n^{-2}}\right)\phi^2 \end{aligned} \tag{4.175}$$

und den Erwartungswert im Sinne der Näherung 4.173

$$E\left[u_n|f\right] \approx \phi^2. \tag{4.176}$$

Natürlich wäre es wünschenswert, wenn jeder Trainingswert u_n ein exakt erwartungstreuer Schätzer wäre. Die Näherung nach Gleichung 4.173 ist aber notwendig, da zur Berechnung von u_n nur der Messwert t_n, sein Fehler s_n sowie die Prognosen μ_f und σ_f des Netzes f zur Verfügung stehen. Insbesondere stehen die Messfehler s_i $(i \neq n)$ der anderen Messpunkte nicht zur Verfügung, da sie sich in der Regel auf andere Messstellen beziehen und eine Bestimmung der fiktiven Wirkung dieser Messungen auf die Stelle des Messwerts t_n nicht möglich ist.

Da die Trainingswerte des Netzes ϕ im Erwartungswert und im Rahmen der genannten Näherung die gesuchte Varianz des regionalen Rauschens liefern, werden sich seine Prognosewerte ebenso verhalten. Um das Netz ϕ einerseits zu einem möglichst robusten Schätzer des regionalen Rauschens zu machen und auch andererseits eine Konfidenz zum geschätzten Rauschen angeben zu können, wird nun der „Fehler“ des Trainingswerts u_n bestimmt:

$$\begin{aligned} VAR[u_n|f] &= E\left[(u_n - E[u_n|f])^2 \mid f\right] \\ &= E\left[\left(\frac{(t_n-\mu_f)^2}{1-\sigma_f^2 s_n^{-2}} - s_n^2 - E\left[\frac{(t_n-\mu_f)^2}{1-\sigma_f^2 s_n^{-2}} - s_n^2 \,\middle|\, f\right]\right)^2 \,\middle|\, f\right] \\ &= \frac{1}{(1-\sigma_f^2 s_n^{-2})^2} E\left[\left((t_n-\mu_f)^2 - E\left[(t_n-\mu_f)^2 \mid f\right]\right)^2 \mid f\right] \\ &= \frac{1}{(1-\sigma_f^2 s_n^{-2})^2} E\left[(t_n-\mu_f)^4 - 2(t_n-\mu_f)^2 E\left[(t_n-\mu_f)^2|f\right] + E\left[(t_n-\mu_f)^2|f\right]^2 \mid f\right] \\ &= \frac{1}{(1-\sigma_f^2 s_n^{-2})^2}\left(E\left[(t_n-\mu_f)^4|f\right] - E\left[(t_n-\mu_f)^2|f\right]^2\right). \end{aligned} \tag{4.177}$$

Da die Größe $t_n - \mu_f$ normalverteilt mit Erwartungswert 0 ist, gilt $E[(t_n-\mu_f)^4] = 3VAR[t_n-\mu_f]^2$, und es folgt weiter

$$\begin{aligned} VAR[u_n|f] &= \frac{1}{(1-\sigma_f^2 s_n^{-2})^2}\left(3E\left[(t_n-\mu_f)^2\right]^2 - E\left[(t_n-\mu_f)^2 \mid f\right]^2\right) \\ &= \frac{2E\left[(t_n-\mu_f)^2\right]^2}{(1-\sigma_f^2 s_n^{-2})^2} \end{aligned} \tag{4.178}$$

und in der Näherung

$$\begin{aligned} VAR[u_n|f] &\approx \frac{2(1-\sigma_f^2 s_n^{-2})^2(\phi^2+s_n^2)^2}{(1-\sigma_f^2 s_n^{-2})^2} \\ &= 2(\phi^2+s_n^2)^2. \end{aligned} \tag{4.179}$$

Die Trainingsfehler des Netzes ϕ werden aufgrund dieser Gleichung als

$$v_n = \sqrt{2}(\Phi^2+s_n^2) \tag{4.180}$$

gewählt, wobei Φ eine a priori Schätzung der Standardabweichung ϕ des regionalen Rauschens ist. An dieser Stelle sollte insbesondere nicht $\Phi = \sqrt{u_n}$ verwendet werden: das Netz ϕ erzeugt nur dann — im Sinne der Näherung — einen Prognosewert mit Erwartungswert ϕ^2, wenn die Trainingsfehler $v_1, \ldots, v_N$ nicht von den Messwerten $t_1, \ldots, t_N$ abhängen. Auch empirische Untersuchungen mit $\Phi = \sqrt{u_n}$ führten zu unplausiblen Prognosen.

Somit sind nun die Trainingsdaten des Netzes ϕ vollständig bestimmt. Um den durch die oben genannte Näherung entstandenen Fehler einzuschätzen, betrachten wir den Erwartungswert des Prognosewerts des Netzes ϕ:

$$\begin{aligned} E[\mu_\phi|f] &= E\left[\left(\sum_{n=1}^{N} v_n^{-2}\right)^{-1} \sum_{n=1}^{N} v_n^{-2} u_n\right] \\ &= \left(\sum_{n=1}^{N} v_n^{-2}\right)^{-1} \sum_{n=1}^{N} v_n^{-2} E[u_n|f]. \end{aligned} \tag{4.181}$$

In diese Gleichung könnten nun die Gleichungen 4.175 und 4.180 eingesetzt werden, der entstehende Ausdruck ist aber sehr kompliziert. Der Übersichtlichkeit und Anschaulichkeit halber werden daher zwei Extremfälle für das a priori Rauschen Φ betrachtet:

1. Fall: $\Phi \to \infty$. Es ist

$$\begin{aligned} E[\mu_\phi|f] &= \left(\sum_{n=1}^{N} (\sqrt{2}\Phi^2)^{-2}\right)^{-1} \sum_{n=1}^{N} (\sqrt{2}\Phi^2)^{-2} E[u_n|f] \\ &= \frac{1}{N} \sum_{n=1}^{N} E[u_n|f] \\ &= \underbrace{\frac{1}{N} \sum_{n=1}^{N} \left(\frac{1 - 2\sigma_f^2 s_n^{-2}}{1 - \sigma_f^2 s_n^{-2}} + \sum_{i=1}^{N} \frac{\sigma_f^4 s_i^{-4}}{1 - \sigma_f^2 s_n^{-2}}\right)}\, \phi^2. \end{aligned} \tag{4.182}$$

   Der unterklammerte Teil liegt nach Lemma 3 im Intervall $[1, 2[$.

2. Fall: $\Phi = 0$. Hier ist

$$\begin{aligned} E[\mu_\phi|f] &= \left(\sum_{n=1}^{N} (\sqrt{2}s_n^2)^{-2}\right)^{-1} \sum_{n=1}^{N} (\sqrt{2}s_n^2)^{-2} E[u_n|f] \\ &= \sum_{n=1}^{N} \frac{s_n^{-4}}{\sum_{j=1}^{N} s_j^{-4}} E[u_n|f] \\ &= \underbrace{\sum_{n=1}^{N} \frac{s_n^{-4}}{\sum_{j=1}^{N} s_j^{-4}} \left(\frac{1 - 2\sigma_f^2 s_n^{-2}}{1 - \sigma_f^2 s_n^{-2}} + \sum_{i=1}^{N} \frac{\sigma_f^4 s_i^{-4}}{1 - \sigma_f^2 s_n^{-2}}\right)}\, \phi^2 \end{aligned} \tag{4.183}$$

   und der unterklammerte Teil liegt nach Lemma 3 im Intervall $]0, 1]$.

Leider ist die Bestimmung von Schranken der Gleichung 4.181 für ein allgemeines Φ sehr kompliziert und würde den Rahmen der vorliegenden Arbeit sprengen. Die Schranken für die beiden Extremfälle legen jedoch nahe, dass für ein einigermaßen gut geschätztes $\Phi \in]0, \infty[$ auch der Erwartungswert $E[\mu_\phi|f]$ nahe bei ϕ^2 liegen wird.

In der Praxis ist dies auch genau der Fall, wenn verschiedene Messstellen Einfluss nehmen. Dazu betrachten wir eine Prognose an einer Stelle x. Diejenigen Messungen, deren Stelle x_n weit weg von x liegt, haben einen vernachlässigbar kleinen Einfluss auf die Prognose an x; dies gilt sowohl für das Netz f als auch für das Netz ϕ. Für die obigen Berechnungen relevant sind dann nur noch diejenigen Messungen in der Nähe von x_n. Bei den konkret vorliegenden Korrosionsdaten wurden für derartige Gruppen von

nahe beieinander liegenden Messstellen meist gleiche oder gleichartige Messgeräte verwendet, sodass die Messfehler s_n innerhalb einer solchen Gruppe gleich groß sind. Wie bereits oben bemerkt und in Abschnitt 4.4.2 beschrieben gilt dann exakt $E[u_n|f] = \phi^2$ und das Netz ϕ berechnet exakt erwartungstreue Prognosen der Varianz des regionalen Rauschens.

4.4.4 Implementierung und empirische Ergebnisse

Die vorliegende Implementierung basiert auf den in Abschnitt 4.4.3 beschriebenen Gleichungen. Bei gegebenen Messwerten $t_1, \ldots, t_N$ mit zugehörigen Messfehlern $s_1, \ldots, s_N$ an den Messstellen $x_1, \ldots, x_N$ wird zunächst ein Netz f mit genau diesen Trainingsdaten trainiert. Bei der Korrosion tritt regionales Rauschen nur in bestimmten Bereichen auf, daher wird das Netz ϕ nur auf Wunsch des Benutzers verwendet. Ist dies der Fall, werden an allen Trainingsstellen Prognosen des Netzes f berechnet und dann die Größen $u_1, \ldots, u_N$ nach Gleichung 4.174 bestimmt:

$$u_n = \frac{(t_n - \mu_f(x_n))^2}{1 - \sigma_f^2(x_n)s_n^{-2}} - s_n^2. \tag{4.184}$$

Die Größe Φ wird global für das Netz geschätzt,

$$\Phi^2 = \max\left(\frac{1}{N}\sum_{n=1}^{N} u_n,\ 0\right), \tag{4.185}$$

sodass dann die Größen $v_1, \ldots, v_N$ nach Gleichung 4.180 bestimmt werden können. Netz ϕ wird nun mit den Messwerten $u_1, \ldots, u_N$, den Messfehlern $v_1, \ldots, v_N$ und den Messstellen $x_1, \ldots, x_N$ trainiert. Anders als Netz f, das asymptotisch linear divergierende Basisfunktionen verwendet, besitzt das Netz ϕ asymptotisch verschwindende Basisfunktionen: das regionale Rauschen der Korrosion wird als regionales Phänomen aufgefasst, außerhalb der vermessenen Region wird a priori kein regionales Rauschen angenommen.

Die Kooperation von Netzen berücksichtigt die einzelnen Fehlerkomponenten. Dies bedeutet, dass verschiedene Netze $f_1, \ldots, f_J$ mit ihren Prognosewerten $\mu_{f1}(x), \ldots, \mu_{fJ}(x)$ und zugehörigen Prognosefehlern $\sigma_{f1}(x), \ldots, \sigma_{fJ}(x)$ wie in Abschnitt 4.1 beschrieben kooperieren. Unabhängig davon kooperieren ebenso die Netze $\phi_1, \ldots, \phi_J$ mit ihren nachbearbeiteten Prognosewerten $\tilde{\mu}_{\phi 1}(x), \ldots, \tilde{\mu}_{\phi J}(x)$ und zugehörigen Prognosefehlern $\tilde{\sigma}_{\phi 1}(x), \ldots, \tilde{\sigma}_{\phi J}(x)$. Für Netze mit regionalem Rauschen ist dabei $\tilde{\mu}_{\phi j}(x) = \max\{\mu_{\phi j}(x), 0\}$ die Prognose des Netzes ϕ_j und $\tilde{\sigma}_{\phi j}(x) = \sqrt{\sigma_{fj}^2(x) + \sigma_{\phi j}^2(x)}$. Mit diesem zusammengesetzten Fehler wird sicher gestellt, dass $\tilde{\sigma}_{\phi j}(x)$ außerhalb des Bereichs der Trainingsdaten des j-ten Netzes groß ist; man beachte, dass aufgrund der asymptotisch verschwindenden Basisfunktionen des Netzes ϕ der einfache Prognosefehler $\sigma_{\phi j}(x)$ ebenfalls asymptotisch verschwindet. Für Netze ohne regionales Rauschen wird $\tilde{\mu}_{\phi j}(x) = 0$ definiert und der Messfehler für die Kooperation mit $\tilde{\sigma}_{\phi j}(x) = \sigma_{fj}(x)$ angenommen.

Nach der Kooperation werden die beiden Fehlergrößen $\sigma_f(x)$ (Prognosefehler für den mittleren wahren Wert) und $\tilde{\mu}_\phi(x)$ (Prognose für die Varianz des regionalen Rauschens) zu einem Gesamtfehler

$$\psi(x) = \sqrt{\sigma_f^2(x) + \tilde{\mu}_\phi(x)}, \tag{4.186}$$

der dem Benutzer angezeigt wird, zusammengefasst.

Die vorhandenen Korrosionsdaten bestanden überwiegend dort, wo regionales Rauschen vorhanden war, aus vergleichsweise wenigen Datensätzen, die zudem in mehr als einem Parameter variierten. Daher wurden zwei künstliche Datenmengen erzeugt, um hier anschaulich das Verhalten des Netzes ϕ und das Zusammenspiel der verschiedenen Fehlerkomponenten beispielhaft und anschaulich darzustellen.

Die Trainingsdatenmenge A besteht aus insgesamt 49 Messungen mit je einem Eingangsparameter aus dem Intervall $[1, 32]$ und einem Ausgangsparameter. Die wahre Verteilung ist durch

$$f(x) = \sin(0,2x) \tag{4.187}$$

$$\phi(x) = \begin{cases} 0,6 & \text{für } x \leq 10 \\ 0,06 \cdot (20 - x) & \text{für } x \in]10, 20[\\ 0 & \text{für } x \geq 20 \end{cases} \tag{4.188}$$

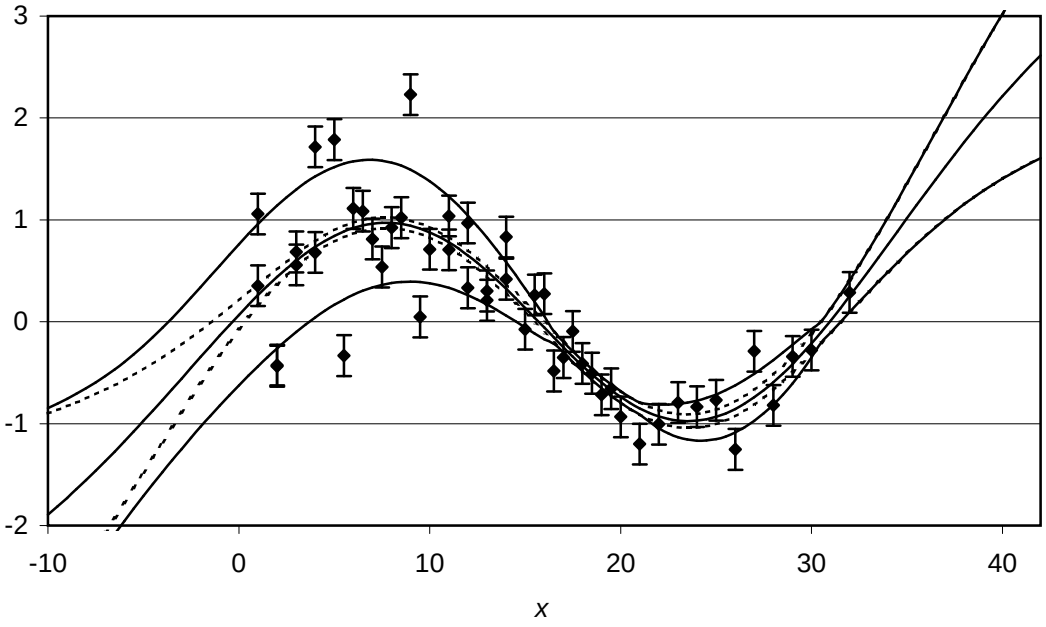

Abbildung 4.8: Ortsabhängiges regionales Rauschen für die künstlichen Trainingsdaten A. Dargestellt sind die Messwerte in Form von Konfidenzintervallen ($t_n \pm s_n$), die Prognosekurve $\mu_f(x)$ (mittlere Kurve), die Prognosefehler $\mu_f(x) \pm \sigma_f(x)$ ohne regionales Rauschen (gepunktete Kurven) und der Gesamtfehler $\mu_f(x) \pm \psi(x)$ (äußere Kurven).

gegeben, das wahre regionale Rauschen war also stetig und stückweise linear. Die Trainingsdaten A sind zusammen mit den Prognosen der Netze f und ϕ in Abbildung 4.8 dargestellt.

Das Rauschen der Trainingsdaten wird vom Netz ϕ gut eingeschätzt. Dass das Netz ϕ im Bereich $16 \ldots 22$ kein regionales Rauschen prognostiziert, ist weniger auf das Verfahren, sondern auf die zufällig geringe Streuung der Trainingsdaten als Stichprobe zurückzuführen. Es ist im übrigen bekannt, dass stochastische Schätzer für die Varianz einer Verteilung bei unbekanntem Erwartungswert sehr große Datenmengen benötigen, um verlässlich zu sein. Diese Schwierigkeit wird hier noch durch die Ortsabhängigkeit der Daten verstärkt.

Man beachte, dass die auf Seite 99 aufgeführten Forderungen 1–4 an ein System zur Prognose von regionalem Rauschen erfüllt sind. Dies wird insbesondere durch die Trainingsdatenmenge B demonstriert, die in Tabelle 4.4 aufgelistet und in Abbildung 4.10 dargestellt ist. Im Bereich $0 \ldots 4$ werden widersprüchliche Daten erkannt und durch ein hohes regionales Rauschen gekennzeichnet. Die Stelle 32 dagegen stellt einen einzelnen isolierten Punkt dar und besitzt somit kein regionales Rauschen. Zwischen diesen beiden Extremen gibt es einen fließenden Übergang.

Abbildung 4.11 zeigt eine Anwendung auf reale Messdaten aus dem Bereich der Korrosion. Man sieht, dass regionales Rauschen erkannt wird, wo es offensichtlich auftritt. Es scheint jedoch, als ob es an manchen Stellen zu klein geschätzt wird, da einzelne Ausreißer immer noch weitab des Fehlerintervalls liegen. Dies liegt jedoch daran, dass diesen einzelnen Ausreißern an jeder betroffenen Stelle mehrere weitere Messungen gegenüber stehen, die unter sich nicht widersprüchlich sind. Das regionale Rauschen ist jedoch als Standardabweichung dieser Verteilung definiert und ist damit deutlich kleiner als etwa die Differenz der beiden Extrema.

Für die Anwendung in der Korrosion stellt sich natürlich hier die Frage, ob dieser Effekt auch gewünscht ist. Gegebenenfalls müssen einige der nicht widersprüchlichen Datensätze weggelassen werden, um das berechnete regionale Rauschen zu erhöhen.

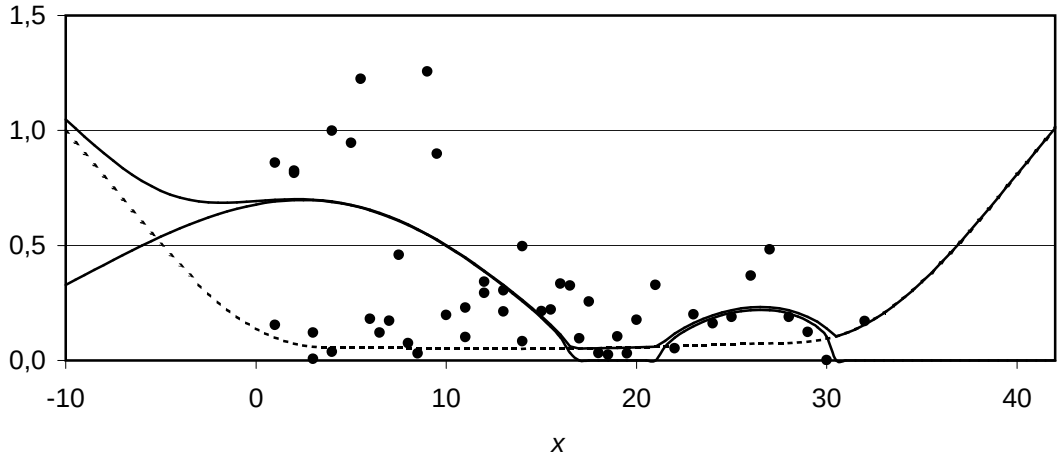

Abbildung 4.9: Detaillierte Darstellung der verschiedenen Fehler zu den Trainingsdaten A aus Abbildung 4.9. Die Punkte stellen die Messpunkte dar, wobei auf der Ordinate die Abweichung zwischen Trainings- und Prognosewert des Netzes f aufgetragen ist: $|t_n - \mu_f(x_n)|$. Die gepunktete Kurve stellt den Prognosefehler $\sigma_f(x)$ ohne regionales Rauschen, die untere Kurve die Standardabweichung des regionalen Rauschens $\sqrt{\mu_\phi(x)}$ und die obere Kurve den Gesamtfehler $\psi(x)$ dar.

x_n	t_n	s_n
0,25	-1	0,2
0,5	1	0,2
1	-1	0,2
2	1	0,2
4	-1	0,2
8	1	0,2
16	-1	0,2
32	1	0,2

Tabelle 4.4: Trainingsdaten B

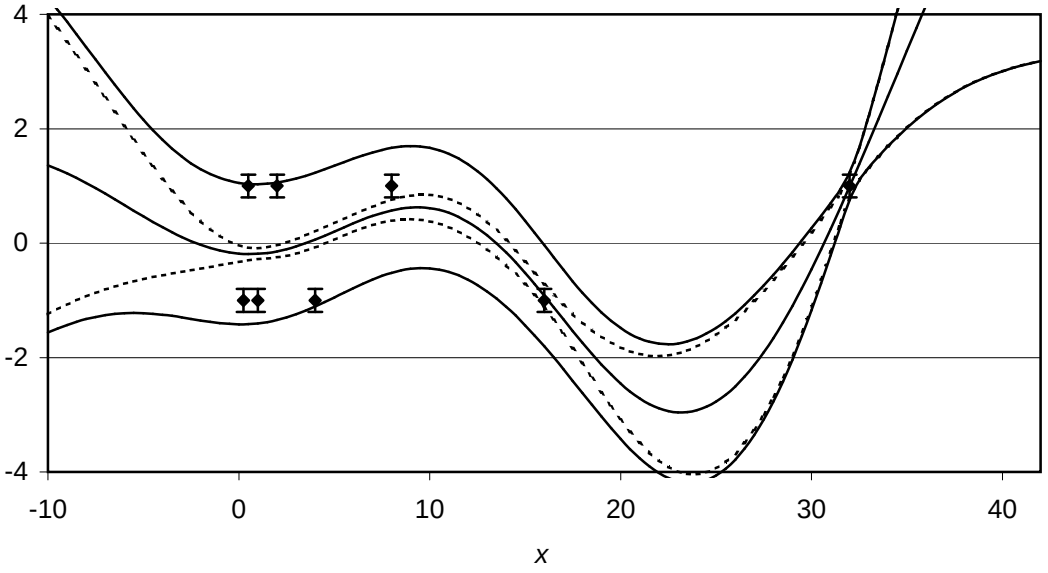

Abbildung 4.10: Ortsabhängiges regionales Rauschen für die Trainingsdaten B. Dargestellt sind die Messwerte in Form von Konfidenzintervallen ($t_n \pm s_n$), die Prognosekurve $\mu_f(x)$ (mittlere Kurve), die Prognosefehler $\mu_f(x) \pm \sigma_f(x)$ ohne regionales Rauschen (gepunktete Kurven) und der Gesamtfehler $\mu_f(x) \pm \psi(x)$ (äußere Kurven).

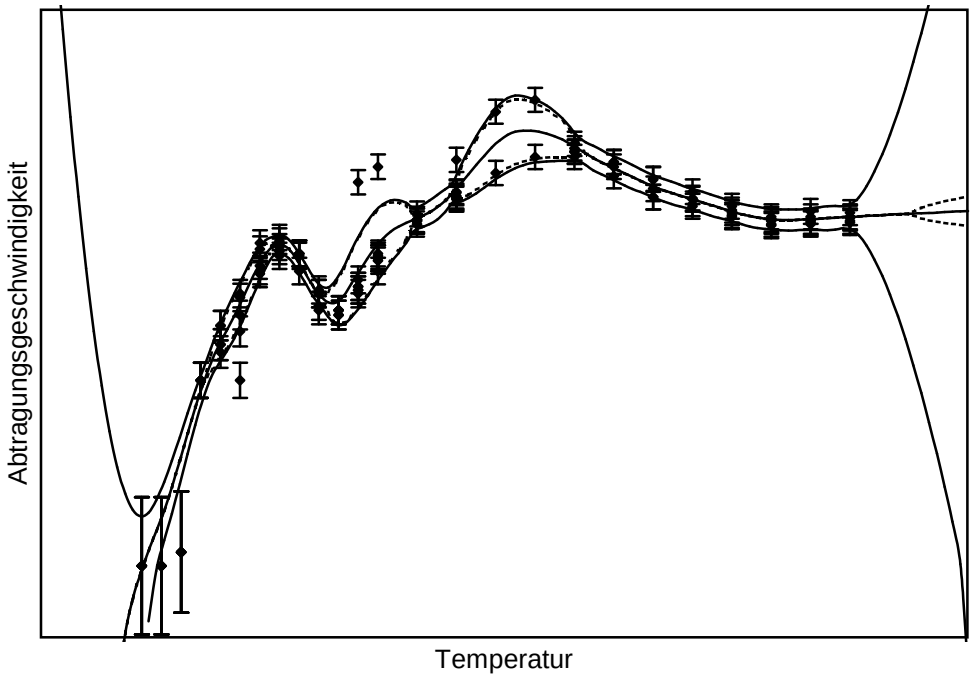

Abbildung 4.11: Regionales Rauschen für reale Trainingsdaten. Es handelt sich um das Korrosionsverhalten eines lochlegierten austenitischen Stahls in korrosivem und saurem Medium. Das regionale Rauschen wird bei diesen Daten durch die Schwankung unbekannter, nicht gemessener Größen verursacht. Man beachte, dass an den meisten Temperaturstellen etwa fünf bis sieben Messungen durchgeführt wurden, bei denen jeweils die meisten Messwerte nahe beieinander liegen.

Kapitel 5

Datenmodellierung

Je mehr über ein Problem bekannt ist, desto eher und besser lässt es sich lösen. Zum Wissen über ein Problem gehören nicht nur die Trainingsdaten selbst, sondern auch Wissen über ihre Interpretation bezüglich der Problemstellung. Das Wissen um die Interpretation der Trainingsdaten kann und sollte in eine geeignete Vorverarbeitung der Daten einfließen, dieses Kapitel widmet sich daher der Methodik einer solchen Vorverarbeitung. Eine Reihe von weiteren Anregungen dazu findet sich etwa in [Bishop], chapter 8, der diesem wichtigen Thema ein ganzes Kapitel einräumt.

Im vorliegenden Anwendungsfall liegt Korrosionswissen in Form von zur Zeit etwa 80 000 Datensätzen in der relationalen Datenbank KISS vor. Diese Datenbank wurde im Wesentlichen zur Dokumentation, Reportgenerierung und einfachen Recherche konzipiert und verwendet. Es liegt jedoch nahe, die Daten darüberhinausgehend miteinander zu verknüpfen und so generalisierende Aussagen zu berechnen. Informationen zu dieser Datenbank finden sich in [Möbius], [Steinmeier], [Azizi] und [Wendler1], Begriffe und Grundlagen zur Korrosion sind in [DIN50900], [DIN50918], [Gräfen] und [Möbius] beschrieben.

Jedes Datenschema zur allgemeinen metallischen Korrosion ist notwendigerweise sehr umfangreich, die Zahl relevanter Größen ist hoch. Die daraus resultierende sehr hohe Dimension des Eingangsraums und die Individualität der einzelnen Parameter unterscheidet das Korrosionsproblem von vielen anderen Problemen, die mit neuronalen Netzen bearbeitet wurden. Anders als bei Bild- und Audiosignalverarbeitungsproblemen oder Zeitreihenanalysen besitzt jeder Parameter der Korrosion eigene Eigenschaften, wie etwa seine physikalische Einheit, seinen Wertebereich oder seinen Einfluss auf die Ausgangsgrößen.

Die in diesem Kapitel beschriebenen Methoden sind natürlich nicht auf Korrosionsprobleme oder diese konkrete Datenbank beschränkt, sondern auch in anderen Datensammlungen einsetzbar. Jedes einzelne Konzept bietet die Lösung zu einem oder mehreren Eigenschaften einer Datensammlung. Insbesondere werden Probleme gelöst, die unter betriebswirtschaftlichen Zwängen entstanden sind, die man also in einer rein wissenschaftlichen Datenbank nicht erwarten würde:

- Es werden nur dort Messstellen erzeugt, wo betriebliche Notwendigkeit besteht, es werden also keine systematischen Messungen durchgeführt.

- Jeder einzelne Datensatz wird so knapp wie möglich beschrieben, denn die Zeit, die ein Benutzer zur Messung und Eingabe von Daten verwendet, verursacht Kosten. Dies führt dazu, dass zwar möglicherweise sehr viele Felder zur Beschreibung eines Datensatzes zur Verfügung stehen, dass aber nur sehr wenige von ihnen auch tatsächlich eingegeben werden. Die Menge der verwendeten Felder kann aber von Datensatz zu Datensatz sehr unterschiedlich sein.

- Geschieht die Dateneingabe durch verschiedene Personen und über einen sehr langen Zeitraum, kann es passieren, dass die problemrelevanten Informationen auf sehr unterschiedliche Art und Weise in die Felder eingegeben werden. Insbesondere Bemerkungsfelder werden oftmals „missbraucht“, um (vermeintliche) Besonderheiten zu charakterisieren.

- Aufgrund betrieblicher Geheimhaltung werden einige Datensätze nicht präzise spezifiziert, trotzdem soll auch ihre Information für das Problem genutzt werden. Als Beispiel kann hier die Beschreibung eines Korrosionsmediums mit der Bezeichnung *Lösungsmittel aus Behälter 23* dienen: natürlich

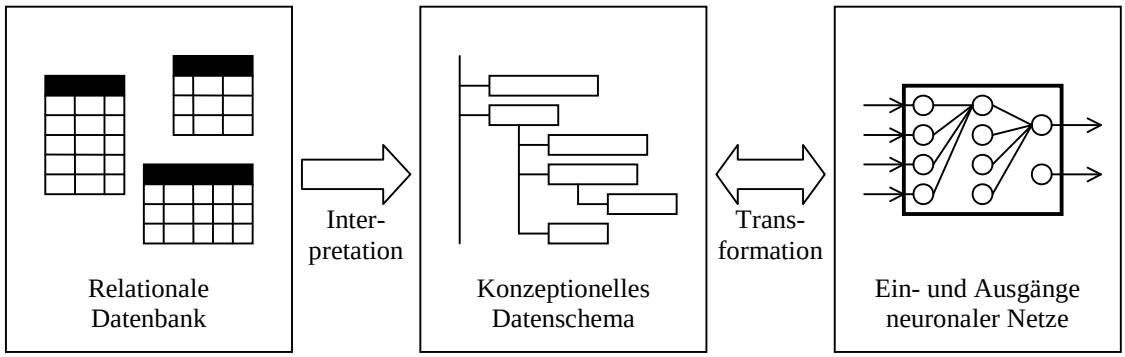

Abbildung 5.1: Grobübersicht über die verwendeten Datenschemata und den Datenfluss im Gesamtsystem.

wäre eine Beschreibung der chemischen Zusammensetzung aus Sicht der Korrosion wünschenswert, um eine Generalisierung über die chemischen Bestandteile zu ermöglichen. Steht dem aber die Geheimhaltung entgegen, so kann man aber immer noch über andere Parameter, wie etwa die Temperatur, generalisieren, wenn entsprechende Messungen vorliegen.

Kapitel 5 beschreibt in allgemeiner Form die Methoden, mit denen Daten mit diesen Eigenschaften auf Daten abgebildet werden, die von neuronalen Netzen mit bayesschen Methoden verarbeitet werden können. Eine detaillierte Beschreibung der konkret verwendeten Abbildung findet sich in [Azizi], Band 2.

5.1 Übersicht

Abbildung 5.1 zeigt die verwendeten Datenformate des gesamten Softwaresystems. Der Inhalt der **relationalen Datenbank** bildet die konkrete Wissensgrundlage, seine Datensätze sind dem System in der Rolle von Trainingsdaten vorgegeben. Natürlich muss die Datenbank, die die ursprünglichen Trainingsdaten enthält, nicht notwendigerweise relational sein. Der Begriff wird hier beispielhaft verwendet, um die Quelle der Messdaten zu bezeichnen. Das relationale Datenschema eignet sich gut zur Eingabe, Darstellung und Recherche von komplexen Daten. In der Praxis ist es allerdings nicht sinnvoll, die Spalten der relationalen Tabellen direkt auf die Ein- und Ausgänge neuronaler Netze abzubilden.

Aus diesem Grund wurde das **konzeptionelle Datenschema** der Korrosion eingeführt. Es ist unabhängig vom relationalen Datenbankschema und beschreibt Datensätze (Korrosionssysteme) anhand der für die Korrosion wesentlichen Kenngrößen. Es vermeidet redundante Darstellungen von Korrosionssystemen, die gleiches Korrosionsverhalten aufweisen. Die Trainingsdaten des relationalen Schemas werden auf entsprechende Daten des konzeptionellen Schemas abgebildet. Diese Abbildung wird im Folgenden hier mit **Interpretation** bezeichnet.

Selbstverständlich ist die konkrete Wahl des konzeptionellen Datenschemas alles andere als eindeutig. In der Praxis orientiert es sich natürlich am relationalen Schema, denn nur dort abgelegte Informationen können im konzeptionellen Schema dargestellt werden. Darüber hinaus muss es auch den Bedürfnissen der täglichen Arbeit der Korrosionsingenieure als Benutzern des Programms angepasst sein. Das konzeptionelle Datenschema wurde während des Projekts häufig verändert, um Wissen über eine verbesserte Interpretation des relationalen Schemas einfließen zu lassen. Dieser Prozess wird auch gegen Ende des Projekts noch nicht beendet sein.

Das konzeptionelle Schema ist vergleichsweise komplex, da es in verschiedenen Aspekten eine hohe Dynamik aufweist; es kann daher nicht direkt auf die Ein- und Ausgänge der neuronalen Netze abgebildet werden. Daher wurde die **Transformation** aller Parameter des konzeptionellen Schemas auf Netzein- und -ausgänge zusammen mit ihm festgelegt. Die Transformation enthält a priori Wissen (Vermutungen, Schätzungen) der Stärke des Einflusses einzelner Parameter auf das Korrosionsverhalten.

Das relationale Datenschema wird hier als vorgegeben angenommen, die Eigenschaften der neuronalen Netze wurden in den vorangegangenen Kapiteln beschrieben. Die weiteren Abschnitte dieses Kapitels

diskutieren daher Methoden des konzeptionellen Schemas, der Interpretation und der Transformation der Daten.

5.2 Das konzeptionelle Schema

5.2.1 Grundtypen von Parametern

Grundlage des konzeptionellen Schemas bilden die Parameter. Es gibt vier verschiedene Grundtypen von Parametern:

- Ein kontinuierlicher Eingangsparameter wie etwa die Temperatur beschreibt eine vorgebbare Eigenschaft eines Korrosionssystems anhand einer Gleitkommazahl, die zusammen mit seiner festgelegten physikalischen Einheit einer physikalischen Messgröße entspricht. Die Menge der gültigen Werte kann eingeschränkt sein, so muss etwa die Temperatur immer größer als der absolute Nullpunkt (-273^{o}C) sein.

- Ein diskontinuierlicher Eingangsparameter wie etwa der Werkstofftyp beschreibt eine vorgebbare Eigenschaft eines Korrosionssystems anhand einer von mehreren Ausprägungen. Der Werkstofftyp etwa besitzt die Ausprägungen *Grundwerkstoff* und *Schweißgut*. Es wird zwischen festen und dynamischen Mengen von Ausprägungen unterschieden: bei den meisten diskontinuierlichen Eingangsparametern ist die Menge möglicher Ausprägungen eher klein und fest durch das konzeptionelle Schema vorgegeben, bei einigen jedoch können jederzeit neue Ausprägungen erzeugt werden, wenn diese benötigt werden. Ein Beispiel für eine dynamische Ausprägungsmenge ist der Produktname des Mediums (Mediumhauptname), der prinzipiell mit jedem neu eingegebenen Korrosionssystem eine neue Ausprägung erhalten kann.

- Ein kontinuierlicher Ausgangsparameter wie etwa die Abtragungsgeschwindigkeit beschreibt eine messbare und sich aus den Eingangsparametern ergebende Eigenschaft eines Korrosionssystems anhand einer Gleitkommazahl für den Wert und einer weiteren Gleitkommazahl für den Fehler. Zusammen mit der physikalischen Einheit des Parameters entsprechen diese beiden Größen einem physikalischen Messwert und seinem Messfehler bzw. einem Prognosewert und seinem Prognosefehler. Außerdem besitzt der Parameter bei der Prognose Angaben zum regionalen Rauschen nach Abschnitt 4.4.3.

- Ein diskontinuierlicher Ausgangsparameter beschreibt eine beobachtbare und sich aus den Eingangsparametern ergebende Eigenschaft eines Korrosionssystems mit Klassifizierungscharakter, die sich für Trainingsdaten und Prognosen unterschiedlich darstellt. Die Behandlung derartiger Parameter beruht auf Abschnitt 4.3.2: bei Trainingsdaten wird der Wert durch die Ausprägung einer beobachtbaren Hinweisklasse D_j beschrieben, bei Prognosedaten wird die Verteilung durch je eine Wahrscheinlichkeitsangabe (Gleitkommazahl) für jede der möglichen wahren Klassen $C_1, \ldots, C_K$ beschrieben.
  In Sinne des in Abschnitt 4.3.2 erwähnten Beispiels des Parameters Korrosionsart würde nun die Anzeige von Korrosionssystemen in Tabellenform für diesen Parameter wie folgt aussehen. Bei Trainingsdaten würde eine Spalte namens „Korrosionsart“ angezeigt, deren Zellen je eine der Ausprägungen *gleichmäßig*, *muldenförmig* oder *lochförmig* enthalten würden. Bei Prognosedaten dagegen würden zwei Spalten namens „Korrosionsart-gleichmäßig“ und „Korrosionsart-ungleichmäßig“ angezeigt, deren Zellen Wahrscheinlichkeitsangaben enthielten. Da sich die Wahrscheinlichkeiten in jeder Zeile stets zu 100% aufsummieren, würde man in diesem Fall dem Benutzer nur eine der beiden Spalten tatsächlich anzeigen.

Die Werte aller Eingangsparameter eines Korrosionssystems beschreiben die Stelle während die Werte und Fehler aller Ausgangsparameter die Trainings- und Prognosewerte und -fehler beschreiben.

5.2.2 Struktur- und abhängige Parameter

Neben diesen Eigenschaften haben die Parameter weitere. Die Eingangsparameter sind hierarchisch angeordnet: es gibt einige Parameter, sogenannte Strukturparameter, von deren Wert die Existenz anderer

Parameter, die sogenannten abhängigen Parameter, in einem bestimmten Korrosionssystem abhängt. Strukturparameter sind dabei grundsätzlich diskontinuierlich. Die Vorteile der hierarchischen Strukturierung in hochdimensionalen Problemräumen wurden bereits in der Literatur beschrieben, etwa bei [RafWil].

Als Beispiel kann die Begasung eines Korrosionssystems dienen: Korrosionssysteme können begast oder unbegast sein. Wenn sie begast sind, dann sollte die Art der Begasung näher spezifiziert werden, denn sie ist korrosionsrelevant; es gibt dann u.a. die Parameter Gasname und Gasmenge. Die Begasung kann nun durch den Strukturparameter Begast mit den Ausprägungen *ja* und *nein* und die abhängigen Parameter Gasname und Gasmenge, die genau im Fall Begast=*ja* existieren, beschrieben werden. Man beachte, dass die Parameter Gasname und Gasmenge immer gemeinsam auftreten.

Natürlich könnte die Begasung auch nicht-hierarchisch mit nur zwei Parametern wie folgt dargestellt werden: die Menge der Ausprägungen des Parameters Gasname wird um die Ausprägung *unbegast* erweitert und im unbegasten Fall wird die Gasmenge mit dem Wert 0 belegt. Die hierarchische Anordnung vermeidet aber gegenüber dieser Lösung unsinnige Parameterbelegungen wie beispielsweise Gasname=*unbegast* und Gasmenge=10m^3/h. Außerdem vereinfacht die hierarchische Anordnung auch die Sicht des Benutzers auf die Daten: sind in einer tabellarischen Darstellung von Korrosionssystemen nur unbegaste Systeme vorhanden, so wird auch nur eine Spalte, nämlich die Spalte „Begast“ angezeigt. Dies fördert sehr die Übersichtlichkeit der Daten, wenn sehr viele Besonderheiten (z.B. Begasung, mechanische, thermische und elektrische Belastungen, usw.) im Datenschema berücksichtigt werden sollen.

Die Begriffe Strukturparameter und abhängiger Parameter beschreiben das relative Verhältnis von Parametern im konzeptionellen Datenschema. Ein bestimmter Parameter kann dabei sowohl abhängig von einem übergeordneten Parameter als auch Strukturparameter für untergeordnete Parameter sein. Alle Parameter, die nicht abhängig sind, bilden die oberste Hierarchieebene, existieren also in jedem Korrosionssystem und werden normale Parameter genannt.

5.2.3 Dynamik der Parameter

Neben den bereits erwähnten dynamischen Ausprägungsmengen einiger diskontinuierlicher Eingangsparameter gibt es noch eine weitere Form der Dynamik im konzeptionellen Schema. Das Korrosionsverhalten in einem Korrosionssystem hängt sehr wesentlich von der Zusammensetzung des Mediums ab. In der chemischen Industrie werden aber sehr viele verschiedene Stoffe verwendet, die Bestandteil des Mediums sein können. Betrachtet man k verschiedene potenzielle Bestandteile, so wäre eine Modellierung des Mediums[1] durch k kontinuierliche Parameter möglich, von denen jeder den Prozentanteil eines Bestandteils am Medium beschreibt. Leider ist die Menge der Bestandteile nicht fest vorgegeben, sondern ergibt sich aus den verwendeten Trainingsdaten: jederzeit können neue Bestandteile in der relationalen Datenbank definiert werden.

Es ist somit notwendig, dynamisch neue Parameter zu erzeugen. Dazu wurde das Konzept der abstrakten Parameter eingeführt: ein abstrakter Parameter enthält alle Eigenschaften eines Parameter mit Ausnahme einer Parameteridentität und einem Parameternamen. Diese Eigenschaften werden erst dann festgelegt, wenn ein abstrakter Parameter mit einem Bestandteil zu einem dynamischen Parameter verbunden werden. Im Sinne des obigen Beispiels existiert also ein abstrakter kontinuierlicher Eingangsparameter „Massenprozent“, aus dem dynamische kontinuierliche Eingangsparameter wie „H_2SO_4-Massenprozent“, „HCl-Massenprozent“ oder „H_2O-Massenprozent“ hergeleitet werden.

Im konzeptionellen Datenschema wird daher jedes Medium durch eine unendliche Menge von dynamischen Parametern beschrieben. Um aber nur die interessanten Parameter verwalten und anzeigen zu müssen, wurde das Konzept des Defaultwerts eingeführt. Zu jeder Sammlung von Korrosionssystemen wird dazu eine endliche Teilmenge aller Bestandteile verwaltet, die mindestens diejenigen Bestandteile enthält, für die mindestens ein Korrosionssystem aus der Sammlung einen Massenprozent-Wert ungleich dem Default, hier 0, besitzt.

Als Beispiel soll folgende Menge von Korrosionssystemen dienen:

[1] Die implementierte Modellierung des Mediums ist komplizierter.

Korrosionssystem	H_2SO_4-MP[2]	HCl-MP	H_2O-MP	...
1	10%	0%	90%	
2	15%	0%	85%	
3	0%	20%	80%	

Obwohl alle drei Korrosionssysteme in der relationalen Datenbank mit nur je zwei Bestandteilen eingetragen sind, müssen hier (mindestens) drei Bestandteile verwaltet und dargestellt werden. Die Ellipse dient lediglich der Erinnerung, dass unendlich viele Spalten konzeptionell vorhanden sind. Dabei werden die Werte für den Parameter HCl-Massenprozent in den Korrosionssystemen 1 und 2 sowie der Wert für den Parameter H_2SO_4-Massenprozent im Korrosionssystem 3 automatisch durch das konzeptionelle Datenschema definiert: es handelt sich um den Defaultwert des abstrakten Parameters Massenprozent.

Softwaretechnisch wird eine Datenstruktur verwendet, die nur diejenigen Werte eines Parameters (einer Spalte) speichert, die vom Defaultwert verschieden ist. Dadurch wird zwar ein wenig mehr Laufzeit beim Zugriff auf einzelne Werte benötigt, der Speicherplatzbedarf sinkt aber enorm: während in der Datenbank insgesamt mehrere Tausend Bestandteile definiert sind, sind in jedem einzelnen Korrosionssystem meist nur zwei oder drei Bestandteile tatsächlich vorhanden.

Man beachte, dass die Konzepte der hierarchischen Strukturierung und der dynamischen Parameter unabhängig voneinander und kombinierbar sind.

5.2.4 Verteilte Werte von Parametern

Bei Trainingsdaten (nicht bei Prognosestellen) ist es möglich, für einen Eingangsparameter, der gemäß des konzeptionellen Schemas in einem Korrosionssystem existiert, nicht einen konkreten Wert, sondern eine Verteilung von Werten anzugeben. Dies ist erforderlich, um fehlende Werte des relationalen Schemas verarbeiten zu können, was in Abschnitt 5.3.1 auf Seite 119 näher erläutert wird. Ein verteilter Wert bedeutet daher, dass der exakte Wert des Parameters nicht bekannt ist, dass er aber in einer bestimmten Verteilung vermutet wird. Diese Verteilung ist für alle Korrosionssysteme, die keinen konkreten Wert besitzen, identisch und durch den Parameter festgelegt; sie wird daher auch Defaultverteilung genannt.

Betrachten wir das Beispiel des Drucks für vier Korrosionssysteme.

Korrosionssystem	Druck ...
1	1 bar
2	verteilt
3	2 bar
4	verteilt

Während die Korrosionssysteme 1 und 3 konkrete Drücke besitzen, sind die Drücke der Korrosionssysteme 2 und 4 identisch verteilt: da die Medien in technischen Anlagen meist unter Überdruck stehen, ist der Druck log-normalverteilt mit den Parametern $\mu = \sqrt{10}$bar und $\sigma = \sqrt{10}$bar, also mit dem Schwerpunkt in der ersten Dekade.

Es gibt nur wenige Parameter, die eine Defaultverteilung erlauben. Es handelt sich dabei um Parameter, die einen nachweisbaren, aber nicht allzu starken Einfluss auf das Korrosionsverhalten haben, und die im relationalen Schema oft nicht angegeben sind.

Die Defaultverteilung kann bei kontinuierlichen Parametern eine völlig beliebige Verteilung von Werten aus dem Wertebereich des Parameters sein. In der Praxis wird meist eine Normal- oder Log-Normalverteilung gewählt. Die Defaultverteilung eines diskontinuierlichen Parameters ist eine beliebige Wahrscheinlichkeitsverteilung über den Ausprägungen des Parameters, in der Praxis meist eine Gleichverteilung.

Verteilte Werte in den Trainingsdaten können zu einer drastischen Erhöhung der Anzahl der zu trainierenden Datensätze der Netze führen, siehe dazu Abschnitt 5.4.7. Dies kann jedoch durch eine Trennung von Korrosionssystemen mit konkreten und mit verteilten Werten vermieden werden.

5.2.5 Nebenbedingungen unter den Parametern

Das konzeptionelle Datenschema besitzt eine vergleichsweise komplexe Struktur. Mit dieser Komplexität soll eine möglichst ein-eindeutige Beschreibung eines Korrosionssystems ermöglicht werden, sodass bereits

[2] MP = Massenprozent

das verwendete Datenschema eine möglichst gute Ausgangsbasis für gute Generalisierungseigenschaften des Gesamtsystems bildet. Leider kann eine ein-eindeutige Beschreibung nicht immer durch die Datenstruktur alleine gewährleistet werden. Es gibt daher zur Zeit zwei Arten von Nebenbedingungen, für deren automatische Einhaltung die Software aktiv sorgt.

Die erste Nebenbedingung besteht im Parameter „Anzahl der Medienbestandteile“, der aus technischen Gründen zur Unterstützung der Einteilung der Experten (Abschnitt 5.2.6) geschaffen wurde. Dieser Parameter wird bei der Interpretation der Trainingsdaten und vor jeder Prognoseanfrage automatisch berechnet und gesetzt. Er ist daher nicht vom Benutzer eingebbar und er wird den Netzen nicht als eigenständiger Eingang zur Verfügung gestellt, ansonsten unterscheidet er sich nicht von den übrigen Parametern des Schemas.

Die zweite Art von Nebenbedingungen ist dagegen wesentlich schwieriger zu handhaben. Sowohl Werkstoffe als auch Medien können über ihre Bestandteile und deren Prozentanteile beschrieben werden, in beiden Fällen müssen sich daher die Prozentanteile zu 100% aufsummieren. Bei der Interpretation der Trainingsdaten werden daher die Legierungs- und Medienbestandteile geprüft und gegebenenfalls durch Multiplikation auf 100% normiert. Bei den Prognoseanfragestellen geht dies nicht so einfach: hier muss der Benutzer einen Legierungs- und einen Medienbestandteil als Rest auszeichnen, der dann vom System automatisch ergänzt wird. Die Software muss dann nur noch die Ungleichungsnebenbedingung „Summe der Bestandteile mit Ausnahme des Rests darf 100% nicht überschreiten“ sicherstellen. Genaueres findet sich in [Wendler2].

5.2.6 Experten und Expertenbereiche

Ein Experte ist eine Einheit, die mit Hilfe von Messdaten trainiert werden und anschließend Prognosen berechnen kann. Dabei handelt es sich bei den Trainings- und Prognosedaten um Daten des konzeptionellen Schemas. Ein Experte besteht aus mehreren Netzen, der Abbildung (Transformation) und Rückabbildung (Rücktransformation) zwischen dem konzeptionellen Schema und den Netzein- und ausgängen, sowie einigen weiteren Informationen[3].

Außerdem ist jedem Experten ein Expertenbereich zugeordnet. Ein Expertenbereich beschreibt eine Teilmenge von Stellen im konzeptionellen Schema, indem er jedem Eingangsparameter eine Wertemenge zuordnet:

- Bei kontinuierlichen Parametern ist diese Wertemenge ein geschlossenes, endliches Intervall, das durch die Intervallgrenzen beschrieben wird.
- Bei diskontinuierlichen Parametern ist die Wertemenge eine nichtleere Teilmenge der Menge der Ausprägungen des Parameters.
- Bei Parametern, die Defaultverteilungen zulassen, kann in der Wertemenge zusätzlich noch das Element *Defaultverteilung* enthalten sein.

Ein abhängiger Parameter existiert im Expertenbereich genau dann, wenn er für mindestens eine Ausprägung aus der Wertemenge seines Strukturparameters existiert.

Bevor die Eigenschaften der Expertenbereiche und ihr Verhältnis zu den Trainingsdaten eingehender beschrieben werden, sollen sie kurz motiviert werden. Die Kooperation von Experten nach Abschnitt 4.1 setzt voraus, dass die Trainingsdatenmengen der beteiligten Experten disjunkt sind. Diese Eigenschaft kann leicht erfüllt werden, indem jedem Experten ein Expertenbereich zugeordnet wird und man von den Expertenbereichen Disjunktheit fordert. Dieses Verfahren unterstützt zusätzlich den Benutzer in dem Bemühen inhaltlich zusammengehörige Bereiche des Eingangsraums in den einzelnen Experten wiederzuspiegeln. Hat man zu einem Experten einen Expertenbereich definiert, ergeben sich nun die Trainingsdaten automatisch: es sind genau diejenigen, die im Expertenbereich enthalten sind. Mit anderen Worten: der Korrosionsingenieur, der die Experten festlegt, ordnet nicht einzelne Korrosionssysteme den Experten zu, sondern beschreibt die Experten über Expertenbereiche. Dass er sich dabei natürlich an den zur Verfügung stehenden Korrosionssystemen orientiert, ist selbstverständlich.

[3] Es handelt sich dabei Informationen, die für den Anwender nützlich sind: die ursprünglichen Trainingsdaten des Experten, das Datum des letzten Trainings, seine Ablage in der Datenbank und eine Bemerkung.

Anschaulich (unter Vernachlässigung der verteilten Werte, der diskontinuierlichen und abhängigen Parameter) ist ein Expertenbereich ein achsenparalleler Quader im hochdimensionalen Raum der Eingangsparameter. Eine Trainings- oder Prognosestelle ist anschaulich ein Punkt in diesem Raum. Man kann nun elementare Operationen auf diesen beiden Arten von Objekten definieren. Sei x eine Stelle und E ein Expertenbereich, dann gilt die Korrespondenz

$$x \in E \tag{5.1}$$

genau dann, wenn die Stelle x im Expertenbereich E enthalten ist. Sei X eine nichtleere Menge von Stellen, dann bezeichnet

$$\text{span}(X) \tag{5.2}$$

den kleinsten Expertenbereich, der alle Stellen aus X enthält. Diese Definitionen sind recht trivial und intuitiv, daher wird hier auf eine formale Einführung verzichtet[4]. Um aber Missverständnissen vorzubeugen, hier ein Beispiel zum Spann.

Stelle	Temperatur	Farbe[5]	Druck	begast	Gasname
x_1	10°C	rot	1 bar	nein	
x_2	30°C	gelb	verteilt	ja	H2
x_3	10°C	grün	2 bar	ja	H2
x_4	20°C	rot	verteilt	ja	verteilt
span($\{x_1, \ldots, x_4\}$)	[10°C, 30°C]	{rot, gelb, grün}	[1bar, 2bar], verteilt	{ja, nein}	{H2}, verteilt

Aufbauend auf diesen Definitionen können nun drei weitere wichtige Definitionen angegeben werden. Seien E_1 und E_2 zwei Expertenbereiche, dann bezeichnet der Ausdruck

$$E_1 \sqcup E_2 \quad := \quad \text{span}\left(\{x \ : \ x \in E_1 \vee x \in E_2\}\right) \tag{5.3}$$

den kleinsten Expertenbereich, der E_1 und E_2 enthält (Vereinigung zweier Expertenbereiche). Die Relation

$$E_1 \sqsubseteq E_2 \quad :\Longleftrightarrow \quad \forall x \in E_1 : \ x \in E_2 \tag{5.4}$$

bezeichnet das Enthaltensein von E_1 in E_2 und die Relation

$$E_1 \perp E_2 \quad :\Longleftrightarrow \quad \neg\exists x : \ x \in E_1 \wedge x \in E_2 \tag{5.5}$$

bezeichnet die Disjunktheit von E_1 und E_2. Die genannten Stellen x sind beliebige Stellen im Eingangsraum, sie sind nicht auf die vorhandenen Trainingsstellen beschränkt.

Die Korrespondenz $\in$, die Funktion span, der Operator $\sqcup$ und die Relationen $\sqsubseteq$ und $\perp$ erinnern sehr an die Mengenalgebra, und tatsächlich gelten eine Reihe von Aussagen auch hier. Seien E_1, E_2 und E_3 Expertenbereiche und x und x_1 Stellen, dann gelten

Ein-Element-Bereich	$x \in \text{span}(\{x_1\})$	$\Leftrightarrow$	$x = x_1$	(5.6)
Assoziativität $\sqcup$	$(E_1 \sqcup E_2) \sqcup E_3$	$=$	$E_1 \sqcup (E_2 \sqcup E_3)$	(5.7)
Kommutativität $\sqcup$	$E_1 \sqcup E_2$	$=$	$E_2 \sqcup E_1$	(5.8)
Selbstvereinigung $\sqcup$	$E_1 \sqcup E_1$	$=$	E_1	(5.9)
Antireflexivität $\perp$	$\neg(E_1 \perp E_1)$			(5.10)
Kommutativität $\perp$	$E_1 \perp E_2$	$\Leftrightarrow$	$E_2 \perp E_1$	(5.11)
Enthalten in Vereinigung	$x \in E_1$	$\Rightarrow$	$x \in (E_1 \sqcup E_2)$	(5.12)
Enthalten in Vereinigung	E_1	$\sqsubseteq$	$(E_1 \sqcup E_2)$	(5.13)
Disjunktheit der Stellen	$x \in E_1 \wedge E_1 \perp E_2$	$\Rightarrow$	$x \notin E_2$	(5.14)
Ausschluss von $\sqsubseteq$ und $\perp$	$\neg(E_1 \sqsubseteq E_2 \wedge E_1 \perp E_2)$			(5.15)
Separation	$E_1 \sqsubseteq E_2 \wedge E_2 \perp E_3$	$\Rightarrow$	$E_1 \perp E_3$.	(5.16)

[4] Sie findet sich aber in der Bayer-internen Dokumentation.

[5] Die Farbe ist kein Parameter der Korrosion und dient hier nur als anschauliches Beispiel.

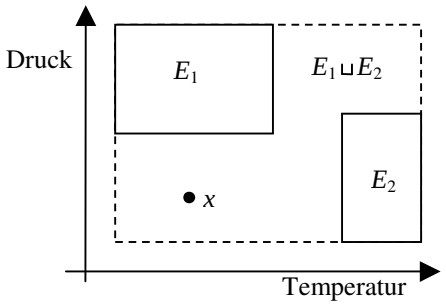

Abbildung 5.2: Vereinigung von Expertenbereichen. Die Stelle x liegt zwar in der Vereinigung $E_1 \sqcup E_2$, jedoch weder in E_1 noch in E_2.

Auf Beweise dieser Aussagen soll hier ebenfalls verzichtet werden, da sie einfach und rein technisch sind.

Einige Operationen und Aussagen der Mengenalgebra gelten jedoch nicht für Expertenbereiche. So gibt es etwa keinen leeren Expertenbereich: in jedem Expertenbereich gibt es mindestens eine Stelle. Aus diesem Grund ist auch kein Schnitt-Operator definiert, denn dieser würde im Fall disjunkter Expertenbereiche einen leeren Bereich bezeichnen. Auch entspricht der $\sqcup$-Operator nicht exakt der Mengenvereinigung: in $E_1 \sqcup E_2$ können Stellen enthalten sein, die zuvor weder in E_1 noch in E_2 enthalten waren, wie Abbildung 5.2 zeigt.

Die verwendete Software implementiert nun Punkte im Eingangsraum des konzeptionellen Schemas (Klasse `KorrosionsDaten`) und Expertenbereiche (Klasse `ExpertenBereich`) als abstrakte Datentypen. Alle oben genannten Beziehungen sind dort implementiert. Da die Expertenbereiche zusammen mit den Experten in der Datenbank gespeichert werden, sind einige Beziehungen sogar zusätzlich in Form von dynamischem SQL implementiert.

Da eine vollständige Abdeckung aller Trainingsdaten durch Experten, also Expertenbereiche, angestrebt wird, da die Expertenbereiche paarweise disjunkt sein müssen, und da nicht alle Experten und Trainingsdaten gleichzeitig im Anwenderclient verarbeitet werden können, wurde die folgende Strategie zur Bearbeitung von Expertenbereichen realisiert. Der Benutzer definiert zunächst einen Expertenbereich E_A als den zu bearbeitenden Bereich (sogenannter Anfragebereich) im konzeptionellen Schema. Nach Ausdruck 5.15 gehört nun jeder vorhandene Expertenbereich in genau eine der folgenden drei Mengen:

$$U_A := \{E_k \text{ aus der DB}: E_k \sqsubseteq E_A\} \tag{5.17}$$

$$V_A := \{E_k \text{ aus der DB}: \neg(E_k \sqsubseteq E_A \vee E_k \perp E_A)\} \tag{5.18}$$

$$W_A := \{E_k \text{ aus der DB}: E_k \perp E_A\}\,. \tag{5.19}$$

Dabei bezeichnet U_A die Menge der Expertenbereiche, die vollständig in E_A liegen, V_A die Menge derer, die teilweise in E_A liegen und W_A die Menge derer, die nicht mit E_A überlappen. Siehe dazu Abbildung 5.3.

Um die Disjunktheitsbedingung aller Expertenbereiche in der Datenbank zu garantieren, genügt es nun alle Expertenbereiche aus $U_A \cup V_A$ im Anwenderclient zu halten, wenn Änderungen nur innerhalb von E_A vorgenommen werden. Mögliche Änderungen können dabei die Erzeugung neuer Expertenbereiche E_k unter der Bedingung $E_k \sqsubseteq E_A$ oder die Änderung von Expertenbereichen $E_k \in U_A$ sein, solange auch nach der Änderung $E_k \sqsubseteq E_A$ gilt. Ausdruck 5.16 stellt dabei die Disjunktheit aller neuen oder geänderten Expertenbereiche mit jedem Expertenbereich aus der Menge W_k sicher, die Disjunktheit mit den Expertenbereichen aus $U_A \cup V_A$ muss natürlich individuell geprüft werden.

Die Änderung von Experten aus V_A ist nur in bestimmten Fällen möglich (z.B. Löschung oder Verkleinerung), daher erlaubt die Software der Einfachheit halber gar keine Änderung. In der Praxis sind in V_A ohnehin selten Expertenbereiche enthalten, denn der Benutzer wählt in der Regel Anfragebereiche, die bereits ein hartes Entscheidungskriterium für die Expertenbereiche enthalten. Ein typisches Beispiel hierfür ist die Anfrage nach allen Salzsäure-Expertenbereichen: demnach würden genau diejenigen Expertenbereiche in V_A enthalten sein, die sowohl zu salzsäurehaltigen als auch zu nicht salzsäurehaltigen Medien Prognosen berechnen können. Ein solcher Experte ist aber in der Praxis wenig sinnvoll.

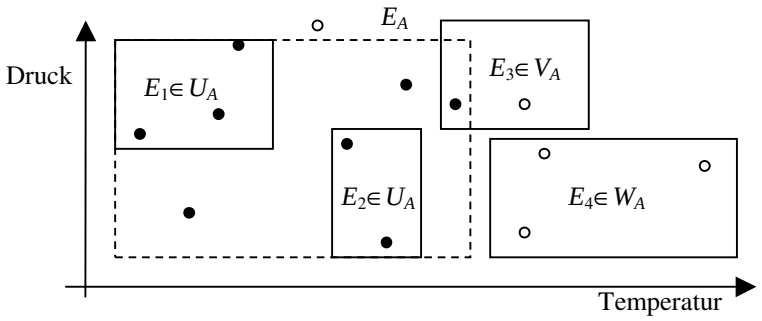

Abbildung 5.3: Beispiel zur Bearbeitung von Expertenbereichen. Der Anfrageexpertenbereich ist gestrichelt gezeichnet, die anderen Expertenbereiche sind in der Datenbank gespeichert und müssen paarweise disjunkt sein. Um sinnvolle Expertenbereiche definieren zu können, werden dem Benutzer auch die Messstellen angezeigt, die hier als kleine schwarze Kreise innerhalb und kleine weiße Kreise außerhalb des Anfragebereichs dargestellt sind.

5.3 Interpretation der Daten

In diesem Abschnitt sollen die Methoden, nach denen die Ursprungsdaten des relationalen Datenbankschemas in Daten des konzeptionellen Schemas abgebildet werden, erläutert werden. Einige von ihnen scheinen sehr trivial oder selbstverständlich, es soll hier aber eine möglichst vollständige Auflistung der verwendeten Methoden präsentiert werden. Natürlich ist die Auflistung keineswegs vollständig im Sinne einer Interpretation beliebiger Schemata.

Außerdem soll an dieser Stelle noch einmal an den weniger naturwissenschaftlich und mehr ingenieurwissenschaftlich/betriebswirtschaftlichen Charakter der vorliegenden Datenbank erinnert werden. Die Festlegung der Interpretation erforderte intensive Recherchen in der Datenbankdokumentation, zahlreiche statistische Auswertungen der Daten und lange Diskussionen mit den Korrosionsingenieuren. Trotzdem hat sich dieser Arbeitseinsatz gelohnt, denn der Gewinn an Wissen über das Problem der Korrosion war groß, gemessen an der verbesserten Generalisierungsfähigkeit des Gesamtsystems.

Man beachte, dass sämtliche beschriebenen Methoden völlig unabhängig von konkreten Trainingsdaten sind.

5.3.1 Interpretation einzelner Felder

Das Feld ist die atomare Informationseinheit des relationalen Schemas, der Wert die des konzeptionellen Schemas. Oftmals können relationale Spalten mehr oder weniger direkt in Parameter des konzeptionellen Schemas abgebildet werden. Dieser Abschnitt beschreibt eine Reihe von Methoden der Interpretation, die in genau dieser Situation auftreten können.

Fehlende Werte Im relationalen Schema kann ein Feld ohne Wert sein, wenn es nicht eingegeben oder automatisch gesetzt wurde. In SQL wird dies durch den speziellen Wert *null* gekennzeichnet. Es gibt nun verschiedene semantische Gründe, warum in einem Feld ein *null*-Wert vorliegt, die voneinander unterschieden werden sollten, um das Feld sinnvoll auf das konzeptionelle Schema abzubilden.

- *Null*-Werte aufgrund von Default-Annahmen. Bei der Eingabe kann Zeit und in der Datenbank Speicherplatz gespart werden, wenn Felder nur dann mit einem konkreten Wert belegt werden, wenn dort eine Besonderheit, also eine Abweichung vom üblichen Wert, vorliegt.
  So wird beispielsweise der relative Untersuchungsort mit den Ausprägungen *im Medium*, *halb im Medium* und *im Dampfraum* oft nicht angegeben, weil die Untersuchung in den meisten Fällen *im Medium* stattgefunden hat. In diesen Fällen kann einfach ein Default für die Spalte definiert werden und diese dann als Parameter in das konzeptionelle Schema übernommen werden.

- *Null*-Werte aufgrund von Nichtwissen. In der betrieblichen Praxis kommt es häufig vor, dass einige Größen eines Datensatzes nicht bestimmt wurden. Dies ist beispielsweise der Fall, wenn Daten aus Literaturquellen eingegeben werden, oder wenn die Datenerhebung erst nach bestimmten Ereignissen (z.B. Schadensfällen) stattfindet. Häufig werden auch nicht alle Parameter gemessen, um kurzfristig Kosten zu vermeiden.
  Hier muss nach der Relevanz der Größe unterschieden werden: wenn sie keinen messbaren Einfluss auf das Phänomen hat, sollte sie nicht in das konzeptionelle Schema übernommen werden. Hat sie dagegen starken Einfluss auf das Phänomen, müssen Datensätze, in denen im relationalen Schema kein Wert angegeben ist und für die auch kein Wert aus anderen Feldern ermittelt werden kann, wegen Unvollständigkeit vom Training ausgeschlossen werden. Hier sollte der Benutzer über dieses Problem informiert werden, damit er gegebenenfalls den Wert ergänzen oder den gesamten Datensatz löschen kann. Hat der Parameter einen messbaren, aber geringen Einfluss auf das Phänomen, so kann auch ein verteilter Wert nach Abschnitt 5.2.4 modelliert werden.

- *Null*-Werte anstelle von vielen Datensätzen. *Null*-Werte können auch absichtlich eingegeben werden, um auszusagen, dass ein bestimmtes Phänomen für beliebige Werte einer physikalischen Größe auftritt. Werden mehrere Spalten so mit *Null*-Werten belegt, können sehr große Teilräume mit einem einzigen Datensatz beschrieben werden. Einem Datensatz mit derartigen *Null*-Werten liegt dann einerseits eine Messung und andererseits das Wissen eines Fachmanns zugrunde. Leider kommt es vor, dass die Unabhängigkeit des Phänomens von der Größe nur für einen bestimmten Wertebereich gilt, was vom Fachmann u.U. nicht berücksichtigt oder gekennzeichnet wurde.
  Theoretisch ist es möglich, Netze direkt mit derart unvollständigen Daten zu trainieren. In der Praxis kann die Entwicklung spezieller Trainingsalgorithmen aber vermieden werden, indem der entsprechende Datensatz expandiert wird: man definiert eine Menge von repräsentativen Stützstellen aus dem Wertebereich und erzeugt für jede Stützstelle einen neuen Datensatz, wobei alle übrigen Felder kopiert werden. Man beachte, dass im Gegensatz zur Expansion von Defaultverteilungen (Abschnitt 5.4.7) hier die Messfehler durch die Expansion nicht verändert werden, da das Phänomen überall im Bereich mit einfacher Konfidenz gemessen wurde. Der Abstand der Stützstellen sollte in Abhängigkeit der Sensitivität (siehe Abschnitt 5.4.2) gewählt werden.

- *Null*-Werte aufgrund von Unsicherheit. Viele Datenbankfelder beschreiben eine nur subjektiv zu bestimmende Eigenschaft des Datensatzes. Manche Menschen tendieren daher dazu, bei einer Datenbankeingabe lieber nichts einzugeben als eine falsche Angabe zu machen. Hat jemand etwa die Farbe *türkis* beobachtet, für die Eingabe aber nur die Auswahl zwischen den Ausprägungen *grün*, *blau* und *rot*, könnte er dazu tendieren, das Feld leer zu lassen.
  Dieser Fall ist nach Abschluss der Eingabe schwierig von den oben genannten Fällen zu unterscheiden. Gelegentlich finden sich Hinweise auf Probleme bei der Eingabe in Bemerkungsfeldern, die natürlich nicht durch die Interpretation automatisch auswertbar sind. Man sollte daher schon bei der Festlegung des relationalen Schemas derartige Probleme erkennen und diese mit den Mitarbeitern, die Daten eingeben, diskutieren.

- *Null*-Werte aufgrund eines unpräzisen Schemas. Diese treten etwa auf, wenn Feldeintragungen nur in einem bestimmten Kontext anderer Felder sinnvoll sind. So macht beispielsweise bei der Begasung das Feld Gasname nur dann einen Sinn, wenn das Feld Gasmenge einen echt positiven Wert besitzt. Ist die Gasmenge dagegen 0, also kein Gas vorhanden, kann das Gas auch nicht benannt werden. Prinzipiell könnten derartige Abhängigkeiten in den Feldern durch Normalisierung des relationalen Schemas eliminiert werden (siehe dazu etwa [Vossen], Kapitel 7). In der Praxis verzichtet man jedoch oft darauf, um das Datenschema und die zugehörigen Eingabemasken einfach zu halten.
  Sind bei einer Spalte einer relationalen Tabelle derartige *null*-Werte zu erwarten, so sollte der Parameter abhängig gemacht werden und nur dann existieren, wenn aus dem relationalen Schema ein konkreter Wert ermittelt werden kann.

Besondere Werte Neben den *null*-Werten gibt es noch weitere Werte, die einer besonderen Interpretation bedürfen. Bei Gleitkommazahl-Feldern hat der Wert 0 oftmals die Bedeutung von *kommt nicht vor*.

Allerdings sollte die Interpretation zwischen *kommt nicht vor* und *unterhalb der Messgrenze* unterscheiden können, wenn dies für die Prognose des Phänomens wesentlich ist, siehe dazu auch Abschnitt 5.3.5.

Analog können auch andere numerische Werte eine bestimmte Bedeutung haben, 100% etwa kann bei Gemischen *nichts anderes* oder *anderes unterhalb der Messgrenze* bedeuten. Liegt die Messgenauigkeit bei einer Einheit, so tendieren manche Schemadesigner dazu, die Angabe *unterhalb der Messgrenze* in eine Art worst-case-Wert von $0,9$ Einheiten zu übersetzen[6]. Die Interpretation sollte hier aber ein besseres Modell verwenden, etwa den Messwert $0,5$ Einheiten mit einem Messfehler von $0,5$ Einheiten.

Auch bei Textfeldern gibt es eine Reihe von Ausprägungen mit besonderer Bedeutung. So taucht etwa die Ausprägung *sonstige* häufig in endlichen Auswahllisten auf, um Datensätze, die eigentlich nicht im Schema darstellbar sind, doch erfassen zu können. Man muss sich aber im Klaren darüber sein, dass *sonstige* eigentlich die Menge aller Ausprägungen bezeichnet, die nicht in der Auswahlliste zu finden sind. Nur solange sich alle diese *sonstigen* Ausprägungen bezüglich des Phänomens gleich verhalten, ist *sonstige* eine Ausprägung wie alle anderen auch. Gegebenenfalls muss *sonstige* in einen verteilten Wert abgebildet werden oder sogar der gesamte Datensatz wegen mangelnder Aussagekraft verworfen werden.

Auch die Ausprägung *unbekannt* taucht gelegentlich in endlichen Auswahllisten auf. Sie ist in der Regel aber äquivalent zu einem *null*-Wert.

Ein/Ausgangsverhalten Ob ein Parameter des konzeptionellen Schemas einen Ein- oder Ausgang eines Experten beschreibt, wurde in der Implementierung danach entschieden, ob er einer physikalischen Ursache oder einer Wirkung entspricht. Der Wert eines Eingangsparameters kann durch den Experimentator mehr oder weniger frei vorgegeben werden, der Wert eines Ausgangsparameters ergibt sich zwangsweise durch das Phänomen (die Messung). Die Korrosion ist vom Wesen her eine Funktion, die den Eingangsparametern einen Wert oder eine stochastische Werteverteilung (regionales Rauschen, Abschnitt 4.4) zuordnet.

Diese eindeutige Unterscheidung von Ein- und Ausgangsparametern führt dazu, dass in der Kooperation jeder Experte die gleichen Ein- und Ausgangsparameter besitzt und daher die Kooperation einfach zu handhaben ist. Es gibt jedoch auch in der Korrosion einige Größen, die sowohl als Eingangs- als auch Ausgangsparameter interpretiert werden können (sie sind jedoch im derzeitigen Schema nicht enthalten). Die metallische Korrosion ist in der Regel ein elektrochemischer Prozess, in dem Metallionenströme die Abtragung des Werkstoffs bewirken. Eine Methode, wichtige Eigenschaften eines Korrosionssystems zu ermitteln, bietet hier die Elektrochemie: der Halbzelle, bestehend aus Werkstoff und Medium, wird eine äußere Spannung aufgezwungen und die resultierende Stromstärke, aus der sich die Abtragungsgeschwindigkeit berechnen lässt, gemessen. Man kann auch umgekehrt vorgehen, dem System einen externen Strom überlagern und die resultierende Spannung messen, siehe dazu [DIN50918]. In beiden Fällen werden also sowohl die Stromstärke als auch die Spannung[7] im relationalen Schema abgelegt. Beide bedingen sich gegenseitig, ohne dass Ursache und Wirkung prinzipiell voneinander unterschieden werden könnten.

Es gibt auch physikalische Größen, die als Zwischengrößen in der Abbildung der Eingangs- auf die Ausgangsgrößen auftreten. In der Korrosion zählt dazu etwa der pH-Wert des Mediums. Einerseits hängt er nur von der Mediumzusammensetzung ab, andererseits enthalten er und der Hauptbestandteil des Mediums alleine bereits genügend Informationen über das Medium, um die Korrosionseigenschaften für die praktische Anwendung oft ausreichend genau zu approximieren[8]. Wenn der pH-Wert nur bei einigen Datensätzen angegeben ist, könnte es sinnvoll sein, zwei Arten von Experten zu verwenden: die erste Art verwendet den pH-Wert als Ausgangsparameter und berechnet ihn aufgrund der Korrosionssysteme, in denen er angegeben ist[9]. Die zweite Art verwendet ihn als zusätzlichen oder die unwesentlichen Medienbestandteile ersetzenden Eingangsparameter zur Bestimmung der Korrosionseigenschaften[10].

Auch außerhalb der Korrosion finden sich Beispiele, bei denen die Bestimmung des Ein/Ausgangsverhaltens nicht trivial ist. Es soll hier auf das medizinische Beispiel aus der Einleitung von Abschnitt 4.3 verwiesen werden, wo die Auswahl der Datensätze über die Ausgangsgrößen erfolgt und die Eingangsda-

[6] Ein konkretes Beispiel liefert der Wert 0,009mm/a der Abtragungsgeschwindigkeit.

[7] tatsächlich die Stromdichte und das Potenzial gegenüber einer Normalelektrode

[8] Bei Medien mit vielen Bestandteilen könnte daher die Anwendung eines Hybridmodells geeignet sein.

[9] Bei wässrigen Lösungen, also den meisten vorkommenden Medien, kann der pH-Wert sogar analytisch berechnet werden.

[10] Die tatsächlich implementierte Interpretation des Mediums enthält die Modellierung des pH-Werts und berücksichtigt Besonderheiten der Beschreibung des Mediums. Sie ist daher noch komplizierter als hier darzustellen sinnvoll wäre.

ten in Abhängigkeit davon gemessen werden. Es liegt dann keine funktionale Beziehung zwischen den Ein- und Ausgangsgrößen (in keiner Richtung) vor, sondern vielmehr ein bestimmter stochastischer Zusammenhang. Alle betrachteten Größen bedingen sich gegenseitig, wie sich etwa Stress und Bluthochdruck gegenseitig beeinflussen können.

Zusammenfassend ist zur Modellierung des Ein/Ausgangsverhalten zu sagen: für die Fragestellungen des Projekts war die „richtige“ Interpretation offensichtlich, für andere, allgemeinere Fragestellungen kann sie sehr kompliziert sein und ggf. komplexe Lösungen erfordern.

Messfehler Physikalische Daten unterliegen immer Messfehlern. Daher können detaillierte Modelle Angaben über die Größe der Messfehler beinhalten. In der vorliegenden Implementierung besitzen nur die Ausgangsparameter Messfehlerangaben. Der Umgang mit diesen Messfehlern an sich wurde in Kapitel 3 allgemein und für besondere Aspekte in den Abschnitten 4.3 und 4.4 erläutert. Hier soll auf die Gewinnung der Messfehlerangaben aus dem relationalen Schema eingegangen werden.

Bei kontinuierlichen physikalischen Größen kann der Messfehler in der Regel leicht abgeschätzt werden, meist kann er auch noch aus den im relationalen Schema vorhandenen Feldern heraus geschätzt werden. Bei vielen Problemen ist es auch ausreichend, für alle Datensätze den gleichen Fehler anzunehmen: beispielsweise $0,5$ Einheiten oder 10% des Messwerts. Man kann zwei Fehlermodelle unterscheiden: implizite Fehlerangaben sind durch die Kenntnis der Messgeräte und -methode gegeben, bei expliziten Fehlerangaben sind diese Angaben direkt in der Datenbank in entsprechenden Feldern abgelegt. Bei der Interpretation von Messfehlern muss berücksichtigt werden, dass der Messfehler des Messgeräts in der Regel eine worst-case Abschätzung darstellt, während der Messfehler des konzeptionellen Schemas die Standardabweichung der Verteilung des wahren Werts beschreibt und somit kleiner ist.

Diskontinuierliche physikalische Größen werden praktisch immer aus kontinuierlichen Größen heraus bestimmt, diese wiederum unterliegen Messfehlern. Oftmals macht man sich die zugrunde liegenden Größen nicht wirklich bewusst, etwa bei der Bestimmung, ob in einem Korrosionssystem Lochfraß vorliegt (siehe Einleitung des Abschnitts 4.3): Lochfraß ist genau dann gegeben, wenn die Tiefe der Vertiefungen in der Werkstoffoberfläche größer ist als deren Durchmesser. In den allermeisten Fällen finden sich entweder keine sichtbaren Vertiefungen oder schmale, tiefe Löcher, sodass die Bestimmung, ob Lochfraß vorliegt, bereits durch kurzes Hinschauen mit hoher Konfidenz beantwortet werden kann. Trotzdem gibt es auch Grenzfälle, in denen genau nachgemessen werden muss, und in denen dann eine Fehlklassifizierung nicht unwahrscheinlich ist.

Als allgemeines Fehlermodell für diskontinuierliche Größen ergibt sich daher eine Funktion, die jeder beobachteten und jeder tatsächlichen Ausprägung eine bestimmte Wahrscheinlichkeit zuordnet. Als einfache Näherung kann bei sehr einfachen Implementierungen angenommen werden, dass die beobachtete und die tatsächliche Ausprägung mit einer bestimmten, hohen Wahrscheinlichkeit gleich sind. Für die vorliegende Implementierung nach Abschnitt 4.3.2 sind die Wahrscheinlichkeiten aber durch eine physikalisch-technische Modellierung des Korrosions- und Messprozesses geschätzt worden ([Azizi]). Sie sind durch die bedingten Wahrscheinlichkeitsverteilungen der wahren Ausprägungen für die im relationalen Schema vermerkten, also beobachteten Ausprägungen gegeben.

Die einzelnen Wahrscheinlichkeiten auch nur annähernd fundiert zu bestimmen ist in der Praxis in der Regel äußerst schwierig. Selbst wenn diese Wahrscheinlichkeiten bekannt wären, ist es fraglich, ob dies zu einer besseren Generalisierung des Gesamtsystems führen würde.

Fehler kontinuierlicher und diskontinuierlicher Größen können nicht nur bei der Messung, sondern auch bei der Übertragung in die Datenbank passieren. Ablese- und Tippfehler werden jedoch völlig anderen Verteilungen unterliegen als Fehler der Messgeräte. Es ist zwar möglich, auch derartige Verteilungen im konzeptionellen Datenschema zu modellieren, etwa durch gemischte Verteilungen, jedoch ist eine algorithmische Umsetzung aufwendig, laufzeitintensiv und wahrscheinlich wenig robust (lokale Extrema der Wahrscheinlichkeitsdichtenfunktion). Zusätzlich besteht meist ein Interesse an korrekten Daten auch im relationalen Schema. Daher ist es die im Projekt verfolgte Strategie, potenziell falsche Datensätze (beispielsweise an stark abweichenden Prognosewerten zu erkennen) manuell zu überprüfen und gegebenenfalls zu korrigieren, und nicht sie zu modellieren.

Natürlich unterliegen nicht nur die Ausgangs-, sondern auch die Eingangsgrößen Messfehlern. Eine entsprechende Modellierung dieser Fehler durch das konzeptionelle Schema kann für beide Parametertypen, kontinuierlich und diskontinuierlich, gleich erfolgen.

Die Verarbeitung von fehlerbehafteten Eingangsparametern durch Netze ist allerdings schwierig, denn sie erfordert Algorithmen, die nicht-identische Verteilungen in allen Eingängen verarbeiten können. Es ist daher einfacher, derartige Fehler als Vergrößerung der Ausgangsfehler zu modellieren. Nehmen wir an, dass es eine wahre Funktion f gibt, die L Eingangsgrößen $x^{(1)}, \ldots, x^{(L)} \in \mathbb{R}$ auf eine Ausgangsgröße $y \in \mathbb{R}$ abbildet: $y = f(x) = f\left(x^{(1)}, \ldots, x^{(L)}\right)$. Seien die Eingangsgrößen mit den Messfehlern (Standardabweichungen) $\Delta x^{(1)}, \ldots, \Delta x^{(L)}$ und die Ausgangsgröße mit dem Messfehler Δy gemessen, dann ergibt sich für den Gesamtfehler aufgrund der bekannten gaußschen Fehlerfortpflanzungsgleichung

$$(\Delta y)_{\text{gesamt}} = \sqrt{(\Delta y)^2 + \sum_{l=1}^{L} \left(\Delta x^{(l)} \frac{\partial f}{\partial x^{(l)}}(x) \right)^2} \tag{5.20}$$

als Approximation für kleine Messfehler.

Es wurden an einigen repräsentativen Beispiel-Korrosionssystemen die Terme Δy und $\Delta x^{(l)} \frac{\partial f}{\partial x^{(l)}}(x)$ für verschiedene Parameter l geschätzt. Dabei wurde für die partiellen Ableitungen die Parametersensitivität (siehe Abschnitt 5.4.2) verwendet. Es ergab sich, dass nahezu ausnahmslos der Summand $(\Delta y)^2$ die Summe in Gleichung 5.20 dominierte, sodass die Messfehler der Eingangsgrößen vernachlässigt werden können.

In einer getrennten Untersuchung (zum numerischen Verhalten der Netze) wurde der Einfluss von künstlich verrauschten Netzeingängen empirisch untersucht. Dabei wurde zunächst jeder Trainingsdatensatz in 10 Datensätze expandiert, die entsprechend mit einem um den Faktor $\sqrt{10}$ vergrößerten Messfehler des Ausgangs trainiert wurden. Anschließend wurde zu allen (transformierten) Eingangsgrößen aller Trainingsdaten je eine Rausch-Zufallszahl addiert, diese Zufallszahlen waren stochastisch unabhängig und normalverteilt mit Erwartungswert 0 und verschiedenen Standardabweichungen:

Rauschen[11]	$\sigma_{\text{w}}^{\text{opt}}$
0	124
0,01	132
0,1	201
0,2	244
0,5	145
1	87
2	13

Leider wurde nicht die Generalisierungsfähigkeit des Netzes bestimmt, aber auch die gefundene optimale Gewichtsregularisierung nach Abschnitt 3.2 gibt Auskunft über eine Veränderung der Prognosen durch das Eingangsrauschen. Die beim Training gefundene optimale Gewichtsregularisierung war für alle Rausch-Standardabweichungen sehr gering ($\sigma_{\text{w}}^{\text{opt}}$ sehr groß). Erst beim stärksten Eingangsrauschen mit einer Amplitude größer den Eingangssensitivitäten, die hier gleich den geschätzten Eingangsmessfehlern waren, war ein signifikanter Effekt auf das Netz zu bemerken. Man beachte, dass alle 42 Eingänge gleichzeitig verrauscht waren und daher der Effekt auf den Ausgang um den Faktor $\sqrt{42}$ größer war als bei nur einem verrauschten Eingang. Insgesamt stützt also auch diese Untersuchung die Behauptung, dass die Messfehler der Eingangsgrößen vernachlässigbar gegenüber denen der Ausgangsgrößen sind.

Der letzte hier angesprochene Aspekt der Messfehler beinhaltet Komponenten des Fehlers, die durch Vereinfachung oder unpräzise Beschreibung entstehen. Dazu zählen etwa Rundungseffekte oder approximative Berechnungen von Größen aus anderen Feldern, wenn die Größen selbst nicht direkt im relationalen Schema eingegeben wurden. Im Sinne verbesserter Generalisierungsfähigkeit des Systems kann es auch sinnvoll sein, inhaltlich verschiedene Datensätze auf die gleiche Beschreibung im konzeptionellen Schema abzubilden, wenn sie bezüglich des Phänomens ähnlich sind. Es kann daher durch verschiedenste Felder bedingte Gründe geben, die zu einer Vergrößerung des/der Ausgangsmessfehler im konzeptionellen Schema führen.

Zusammenfassungen von Ausprägungen Wie im vorigen Absatz bereits erwähnt, kann es günstig für die Generalisierungsfähigkeit des Gesamtsystems sein, wenn mehrere Ausprägungen eines Feldes

[11] Standardabweichung des künstlichen Rauschens der Eingänge

des relationalen Schemas zu einer Ausprägung des konzeptionellen Schemas zusammengefasst werden. Dies sollte dann geschehen, wenn die ursprünglichen Ausprägungen bezüglich des zugrunde liegenden Phänomens äquivalent sind.

Sind die ursprünglichen Ausprägungen nicht ganz äquivalent, so können sie getrennt beibehalten werden; die Ähnlichkeit kann dann durch die Umsetzvektoren (siehe Abschnitt 5.4.3) realisiert werden. Auch so erhält man eine gute Generalisierungsfähigkeit. Der Übergang von äquivalenten zu ähnlichen Ausprägungen kann also fließend realisiert werden.

5.3.2 Interpretation von Wertemengen und dynamischen Daten

Das konzeptionelle Schema sollte (wie das relationale Schema eigentlich auch) so gestaltet sein, dass möglichst wenige logische/technische Abhängigkeiten zwischen den Parametern (Feldern) bestehen. Dies ist jedoch nicht immer und in allen Unterstrukturen möglich.

Einige Größen werden für einen Datensatz nicht durch eine Zahl oder eine einfache Ausprägung, sondern durch einen komplexeren Datentyp dargestellt. Das relationale Schema stellt in diesem Fall eine bestimmte Menge von Feldern für jeden Datensatz zur Verfügung, es kann sich dabei entweder um eine feste oder eine dynamische Anzahl von Feldern und Werten handeln.

Die Interpretation einer Wertemenge und ihre Darstellung im konzeptionellen Schema führt dann zu einer guten Generalisierung des Systems, wenn im Sinne des Problems ähnliche Wertemengen auf eine ähnliche Beschreibung im konzeptionellen Schema abgebildet werden. Es ist daher oft nicht sinnvoll, den der Wertemenge zugrunde liegenden Datentyp unverändert ins konzeptionelle Schema zu übernehmen. Meist ist es günstiger, diejenigen Features zu ermitteln, die den Datensatz problemangemessen beschreiben, dies wird im weiteren Verlauf dieses Abschnitts an konkreten Datentypen diskutiert.

Intervalldaten Für kontinuierliche Größen sind mögliche Wertemengen oft Intervalle, das relationale Schema enthält in diesem Fall dann zwei Spalten namens „von“ und „bis“. Die Darstellung von Einzelwerten kann unter Umständen einen Spezialfall darstellen (etwa „von“ = Einzelwert, „bis“ = *null*), sodass eine einfache Konvertierung nötig sein kann. Sinnvoll kann auch eine automatische Überprüfung der Bedingung „von“ $\leq$ „bis“ sein, um unplausible Datensätze manuell zu kontrollieren.

Die korrekte Interpretation des Intervalldatentyps hängt von der Bedeutung der Größe ab und ist teilweise den *null*-Werten des Abschnitts 5.3.1 ähnlich.

- Drückt das Intervall eine zeitliche oder räumliche Schwankung der Größe aus, so ist die direkte Beschreibung durch die Intervallgrenzen zwar möglich, allerdings oft nicht optimal. Ist die Ausgangsgröße innerhalb des Intervalls mit hinreichender Genauigkeit linear, so reicht der Mittelwert, („von“ + „bis“)/2 zu einer guten Beschreibung aus. Ist sie hochgradig nicht-linear, so können andere berechnete Größen geeignet sein, wie etwa der Mittelwert und die Intervallbreite oder der worst-case Wert (z.B. die obere Intervallgrenze) und die Intervallbreite.
  Kann man weiter Aussagen über die Verteilung der Schwankung machen (kleinere Werte waren beispielsweise häufiger), so könnte etwa ein gewichteter Mittelwert, $\alpha \cdot$„von“$+(1-\alpha)\cdot$„bis“, $\alpha \in [0,1]$, verwendet werden.

- Beschreibt das Intervall einen zwar konstanten, innerhalb des Intervalls liegenden, ansonsten aber unbekannten Wert, sollte eine Verteilung modelliert werden. Wie bereits weiter oben diskutiert, ist die nachfolgende Verarbeitung derartiger, für jeden Datensatz anders gearteter Verteilungen schwierig. Daher sollte der Datensatz bei der Interpretation expandiert, d.h. durch mehrere Datensätze mit vergrößerten Ausgangsmessfehlern ersetzt werden. Die Expansion kann allerdings ein Problem darstellen, wenn mehrere Eingangsgrößen Verteilungen aufweisen, siehe dazu Abschnitt 5.4.7.
  Die Expansion kann trivial sein, d.h. ein Datensatz wird auf nur einen Datensatz abgebildet, wenn die Breite der Verteilung bei allen Daten so klein ist, dass kein Effekt auf die Ausgangsgrößen spürbar ist. Die Intervallinformation ist dann bereits im relationalen Schema (für die Verarbeitung durch neuronale Netze) unnötig und könnte durch nur einen einzelnen Wert ersetzt werden.

- Man kann physikalischen Größen Intervallwerte zuordnen, um auszudrücken, dass die beobachteten Ausgangsgrößen für alle Werte aus dem Intervall gelten. Das Intervall entspricht dann vielen

Messungen und kann durch Expansion des Datensatzes bei unveränderten Ausgangsmessfehlern abgebildet werden.

Mengen Der komplexe Datentyp einer Menge kann im relationalen Schema leicht durch eine 1:n-Beziehung zwischen der Tabelle, die in jeder Zeile einen Datensatz enthält, und einer weiteren Tabelle realisiert werden. Neben dieser Darstellung können Mengen auch durch eine fixe Anzahl von booleschen Werten oder durch lange Textfelder, die als Aufzählung interpretiert werden können, beschrieben werden.

Ein Beispiel für diesen Datentyp ist durch die Erscheinungsformen der Korrosion, eine Ausgangsgröße, gegeben: die Korrosion kann etwa aus einem gleichmäßigen Flächenabtrag bestehen, der zusätzlich noch von Lochfraß überlagert wird. Eine sinnvolle Interpretation der relationalen Spalte Erscheinungsform ist etwa die Aufteilung ihrer möglichen Ausprägungen auf mehrere Ausgangsparameter[12]:

relationale Spalte Erscheinungsform	Parameter Flächenabtrag	Parameter Lochfraß
gleichmäßiger FA[13]	*gleichmäßig*	
ungleichmäßiger FA	*ungleichmäßig*	
Default	*kein*	
Lochfraß		*ja*
Default		*nein*

Die Default-Eintragungen werden so verwendet, dass jeder Parameter immer einen definierten Wert besitzt: ist beispielsweise *gleichmäßiger FA* die einzige eingetragene Erscheinungsform, so besitzt der Parameter Lochfraß den Wert *nein*. Durch diese Interpretation können auch inhaltlich unplausible Datensätze entdeckt werden, etwa wenn für die Erscheinungsform sowohl *gleichmäßiger FA* als auch *ungleichmäßiger FA* angegeben ist. Die Interpretation kann sehr schwierig werden, wenn die relationale Ursprungsspalte sehr viele Ausprägungen zulässt: es muss sorgfältig überlegt werden, welche Kombinationen erlaubt und welche unplausibel sind, was in der Praxis schwierig ist und in der algorithmischen Umsetzung der Interpretation zu sehr umfangreichen Fallunterscheidungen führen kann.

In bestimmten Fällen ist es notwendig, dass die Elemente der Menge noch durch weitere Größen näher beschrieben werden. Ein Beispiel hierfür stellt die Beschreibung des Mediums als Gemisch verschiedener Bestandteile dar. Für jeden Bestandteil existiert daher ein diskontinuierlicher dynamischer Parameter des konzeptionellen Schemas, der angibt, ob der Bestandteil in dem konkreten Medium enthalten ist. Wenn er enthalten ist, existiert ein kontinuierlicher abhängiger Parameter, der seinen Prozentanteil am Medium angibt (warum diese spezielle Art der Darstellung gewählt wurde, wird in Abschnitt 5.3.5 dargelegt). Man beachte, dass die Menge selbst nur anhand der diskontinuierlichen Enthalten-Parameter beschrieben wird. Anhand des Bestandteils identifiziert ein Enthalten-Parameter alle von ihm abhängigen Parameter, die sich wiederum auf den gleichen Bestandteil beziehen.

Gerichtete Listen Fügt man dem Datentyp Menge weitere Struktur in Form der Reihenfolge seiner Elemente hinzu, erhält man den Datentyp gerichtete Liste.

Hier kann die Folge der Fertigungsschritte eines Werkstücks als Beispiel dienen. Die Fertigungsschritte *Beizen*, *Walzen*, *Schweißen* und *Sandstrahlen* können jeder für sich durchgeführt oder nicht durchgeführt werden. Die Reihenfolge ist prinzipiell beliebig und hat Einfluss auf die endgültige Oberflächenbeschaffenheit, also auch auf das Korrosionsverhalten. Die spezielle Schwierigkeit hier liegt in der Bestrebung, eine Interpretation und ein Schema zu finden, das einerseits eine statische Beschreibung für die konkret vorhandenen Datensätze liefert, das andererseits aber die Ähnlichkeit von Fertigungsschrittfolgen auch in der statischen Beschreibung wiederspiegelt.

Eine Möglichkeit der Interpretation ist, auf die Reihenfolge zu verzichten und die Liste als Menge zu betrachten. Man würde dann entweder die Reihenfolgeinformation als wenig korrosionsrelevant vernachlässigen oder immer eine bestimmte feste Reihenfolge implizit annehmen.

Eine andere Möglichkeit besteht darin, Regeln zu finden, die tatsächlich vorkommende Reihenfolgen beschreiben (etwa *Walzen* kommt immer vor *Beizen*). Darauf aufbauend kann man dann eine Superliste

[12] Die implementierte Modellierung und Interpretation der Erscheinungsform ist umfangreicher.
[13] FA = Flächenabtrag

definieren, die alle möglichen Listen enthält, die den Regeln genügen. Das Schema beschreibt dann nur noch die endliche Menge der Fertigungsschritte, die tatsächlich durchgeführt wurden. Als Superliste wäre etwa denkbar: *Walzen - Sandstrahlen 1 - Schweißen - Beizen - Sandstrahlen 2*; ein Werkstück, das zunächst gesandstrahlt und dann geschweißt wurde, erhielte dann als Wert die Liste *nein - ja - ja - nein - nein*.

Problematisch ist hier, dass zu wenige Regeln die Superliste sehr verlängern, dass aber zu restriktive Regeln dazu führen können, dass einzelne Datensätze den Regeln nicht entsprechen und somit nicht mehr durch die Superliste dargestellt werden können. In bestimmten Fällen ist auch die Abbildung der Liste in die Superliste nicht eindeutig. So ist im obigen Beispiel nicht klar, welcher der Fertigungsschritte *Sandstrahlen 1* und *Sandstrahlen 2* mit der Ausprägung *ja* belegt werden soll, wenn *Sandstrahlen* das einzige Element der Ursprungsliste ist. Man beachte dabei, dass die Interpretation die Ähnlichkeit der Datensätze beibehalten soll.

In der vorliegenden Implementierung wurden die Fertigungsschritte in vier Kategorien eingeteilt. Bei drei Kategorien wird nur der zeitlich letzte Fertigungsschritt in einem Parameter dargestellt, da die jeweiligen Kategorieeigenschaften in guter Näherung nur von ihm abhängen. Die Fertigungsschritte der vierten Kategorie werden als Menge interpretiert.

Ungerichtete Listen Ist eine gerichtete Liste zu ihrer Umkehrung, also der Liste mit gleichen Elementen aber umgekehrter Reihenfolge, äquivalent, so ist der zugrunde liegende Datentyp eine ungerichtete Liste. Die Interpretation sollte dabei nach Möglichkeit von der Reihenfolge abstrahieren und eine Liste und ihre Umkehrung auf den gleichen Datensatz des konzeptionellen Schemas abbilden.

In der Werkstofftechnik kann dies etwa bei der geometrischen Anordnung, etwa aneinander geschweißte Rohrstücke, von unterschiedlichen Werkstoffen zu einem Werkstoffsystem sein[14]. Bei drei Werkstoffen ist daher entscheidend, welcher Werkstoff in der Mitte verwendet wurde, die beiden Enden sind jedoch vertauschbar.

Eine adäquate Interpretation ist hier schwierig. Fordert man tatsächlich die Umkehrung der Reihenfolge als Invariante der Interpretation, so muss eine Ordnungsrelation über den Werkstoffen definiert werden: ist der erste Werkstoff „größer“ als der letzte Werkstoff der Liste, so wird ihre Reihenfolge umgekehrt. Werkstoffe besitzen jedoch keine natürliche Ordnungsrelation, mögliche praktische Relationen wären daher die lexikographische Ordnung ihrer DIN-Bezeichnungen oder eine durch ihre Legierungsbestandteile gegebene Ordnung (1. Schlüssel Chrom, 2. Schlüssel Eisen, ...). Diese beiden Ordnungen berücksichtigen aber leider nicht die Ähnlichkeit zweier Werkstoffe.

Man beachte auch, dass die künstliche Ordnung der Liste im konzeptionellen Schema eine Nebenbedingung darstellt, die auch bei Prognosen einzuhalten ist. Es kann daher auch überlegt werden, ob man die Reihenfolge im konzeptionellen Schema freilässt und sie erst während der Transformation auf die Netzeingänge normiert.

Andere Formen Prinzipiell kann neben Mengen und Listen jeder Datentyp, der andere Datentypen enthält, dynamische Parameter oder Gruppen von Parametern beschreiben. Eine generische Interpretation kann nicht angegeben werden, in der Regel müssen auch Kompromisse zwischen Beibehaltung der Ähnlichkeit, Vermeidung von Nebenbedingungen und praktischer Handhabbarkeit (Benutzerfreundlichkeit) getroffen werden.

Es ist auch möglich, dass Parametergruppen wiederum dynamisch andere Parametergruppen enthalten. Dies ist beispielsweise nötig, wenn das Werkstück nicht nur entlang des Mediums sondern auch noch parallel zum Medium aus verschiedenen Werkstoffen besteht, etwa bei Korrosionsschutzbeschichtungen. Abbildung 5.4 zeigt ein Beispiel.

5.3.3 Kombinationen von Feldern

In komplexen Datenbanken kommt es oft vor, dass Informationen auf verschiedene Art und Weise angegeben werden können. Die dazu zur Verfügung stehenden Felder sind genau genommen redundant, erhöhen aber den Komfort des Benutzers bei der Eingabe.

[14] Die geometrische Anordnung verschiedener Werkstoffe in einem Werkstoffsystem wird in der aktuellen Implementierung nicht berücksichtigt.

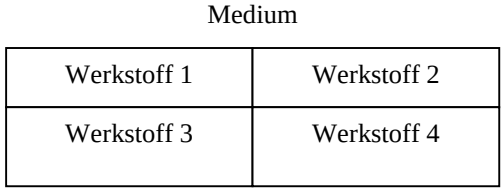

Abbildung 5.4: Aufbau eines komplexen Werkstücks im Korrosionssystem. Die Werkstoffe 1 und 2 werden in direktem Kontakt vom Medium angegriffen, während die Werkstoffe 3 und 4 zunächst geschützt sind.

Im Sinne einer guten Generalisierungsfähigkeit sollten die Felder wieder zu einer nicht-redundanten Darstellung im konzeptionellen Schema kombiniert werden. Dies soll hier am Beispiel der Werkstoffbeschreibung diskutiert werden, die vereinfacht dargestellt wird.

Der Werkstoff eines Korrosionssystems wird anhand der Prozentanteile seiner Legierungselemente beschrieben, da so die Ähnlichkeit zweier Werkstoffe sehr gut dargestellt wird. Die Angabe von Legierungsanteilen ist allerdings optional und wird nur dann durchgeführt, wenn das konkrete Werkstück des Korrosionsversuchs analysiert wurde. Es handelt sich dann um sogenannte Ist-Werte.

Ist-Werte werden eher selten angegeben, häufiger wird der Werkstoff nur über seinen Hauptnamen (ein eindeutiger, normierter Name für Werkstoffe) beschrieben. Dieser Hauptname kann dann über eine weitere Tabelle in seine Legierungsanteile übersetzt werden, wobei es sich dann aber um Soll-Werte handelt. Die Soll-Werte sind genau genommen Intervalle, also herstellungsbedingte Bandbreiten, und beschreiben daher das tatsächlich verwendete Werkstück nicht exakt. Die Abweichungen zwischen Ist- und Soll-Werten kann durchaus Einfluss auf das Korrosionsverhalten haben.

Die Interpretation geschieht nun wie folgt. Sind Ist-Werte vorhanden, werden diese verwendet. Andernfalls werden die Soll-Werte verwendet, wenn diese vorhanden sind. Sind auch diese nicht verfügbar, werden die Mittelwerte aller Ist-Werte von Werkstoffen anderer Korrosionssysteme mit gleichem Werkstoff-Hauptnamen verwendet.

Diese Reihenfolge drückt die erwartete Genauigkeit der Angaben aus: die Ist-Werte sind am präzisesten, danach folgen die Soll-Werte. Nur wenn diese auch nicht vorhanden sind, werden vergleichbare Ist-Werte anderer Korrosionssysteme verwendet, um wenigstens ungefähre Werte verwenden zu können.

5.3.4 Daten außerhalb des Schemas

Auch sehr umfangreiche und komplexe Schemata können nicht garantieren, dass sie alle für die Beschreibung des Phänomens wichtigen Parameter enthalten. In der Praxis kommt es aber gelegentlich vor, dass man Datensätze abgelegen möchte, die durch das Schema nicht hinreichend genau beschrieben werden können. Aus diesem Grund finden sich in den Schemata häufig Felder für Bemerkungen oder Kommentare, in die dann Werte eingetragen werden können, für die eigentlich eigene Spalten bzw. Parameter existieren sollten.

Zu viele Spalten bzw. Parameter führen allerdings zur Unübersichtlichkeit des Schemas, was wiederum Fehleingaben begünstigt. Außerdem wird dadurch ein sehr großer Raum aufgespannt, der dann kaum durch Daten gefüllt wird. Man muss also bei der Festlegung des konzeptionellen Schemas entscheiden, welche Parameter in der Praxis benötigt werden und auf welche Sonderfälle verzichtet werden kann. Wenn beispielsweise nur drei der Korrosionssysteme, die für ein Training zur Verfügung stehen, elektrochemisch belastet wurden, ist es sinnvoll hier auf die Parameter zur Beschreibung der elektrochemischen Belastung zu verzichten und die genannten drei Korrosionssysteme vom Training auszuschließen. Es dürfte von wenig praktischem Nutzen sein, einen speziellen Experten für elektrochemische Belastungen zu erzeugen, wenn ihm lediglich drei Trainingsdaten zur Verfügung stehen.

Entscheidende Bedeutung bekommt daher während der Interpretation der Daten die Untersuchung, ob ein Datensatz ausreichend im konzeptionellen Schema dargestellt werden kann. Verallgemeinernd gibt es nachfolgende Gründe, die dazu führen sollten, dass ein Datensatz nicht interpretiert, also nicht zum Training verwendet wird.

- Im relationalen Schema finden sich Werte in Feldern, zu denen es keine Entsprechung im konzeptionellen Schema gibt (Beispiel wie oben: elektrochemische Belastung). Der Datensatz stellt einen Sonderfall dar.

- In einem Bemerkungsfeld werden wesentliche Angaben zum Phänomen gemacht, die keine Entsprechung im konzeptionellen Schema haben. Der Datensatz stellt auch hier einen Sonderfall dar, der praktische Unterschied ist jedoch, dass derartige Datensätze schwer zu erkennen sind, da Bemerkungsfelder praktisch nicht automatisch untersucht werden können.

- Die Werte des Datensatzes verletzen eine der oben genannten Plausibilitätsbedingungen. Bei der Fortentwicklung einer Datenbank über Jahre hinweg, die etwa Änderungen im Schema mit sich bringt, treten ungültige Datenbankzustände leider gelegentlich auf. Auch ungeschulte Mitarbeiter stellen eine Quelle derartiger Fehler dar. Hier kann es sinnvoll sein, den Datensatz sicherheitshalber wegzulassen, er sollte aber manuell überprüft werden.

5.3.5 Problemorientierte Modellierung: Heuristiken

Es ist durchaus sinnvoll, bestimmte Eigenschaften der modellierten Wirklichkeit, die über die reine Beschreibung hinausgehen, direkt in das Modell einfließen zu lassen.

Als Beispiel können hier die Eigenschaften des Mediums eines Korrosionssystems gelten. Fügt man einem Medium eine winzige Spur eines bestimmten Stoffes hinzu, können sich seine Eigenschaften dramatisch verändern, wie etwa beim Salz in einer Suppe[15]. Ist die Spur so gering, dass sie mit den benutzten Messverfahren nicht gemessen werden kann, oder ist der genaue Anteil am Medium auch nicht relevant, so wird der Anteil des Spurenbestandteils in der Datenbank mit 0 eingetragen.

Ein geeignetes konzeptionelles Schema verwendet hier zwei Parameter für jeden Bestandteil zur Beschreibung des Mediums: ein diskontinuierlicher Strukturparameter mit den Ausprägungen *ist enthalten* und *ist nicht enthalten* und die kontinuierliche Angabe des Prozentanteils als abhängiger Parameter. Somit können sowohl das Nichtvorkommen vom Vorkommen als Spur, als auch eine Spurenkonzentration von einer höheren Konzentration unterschieden werden. Diese Modellierung ist durch die Beobachtung begründet, dass bestimmte Eigenschaften des Mediums als Funktion des Anteils eines Bestandteils bei 0 eine Unstetigkeit (oder zumindest eine extreme Änderung) aufweisen. Dies ist nichts anderes als eine Heuristik des Phänomens, die bereits im konzeptionellen Schema realisiert wird.

5.3.6 Die Rückabbildung vom konzeptionellen in das relationale Schema

Bei der Bearbeitung von Expertenbereichen nach Abschnitt 5.2.6 und beim Training von Experten müssen zu einem vorgegebenen Expertenbereich alle Datensätze ermittelt werden, die in diesem enthalten sind.

Da die Interpretation eine sehr umfangreiche und komplizierte Abbildung darstellt, kann ihre Umkehrabbildung nicht effizient (in SQL) realisiert werden. Probleme bereiten insbesondere Umrechnungen, die mehrere Felder des relationalen Schemas miteinander kombinieren, etwa die Normierung der Prozentanteile der Medien- und Legierungsbestandteile, sodass deren Summe exakt 100% beträgt. Auch die Auflösung von Werkstoffbezeichnungen in ihre Legierungsbestandteile anhand von externen Tabellen ist kaum effizient invertierbar.

Daher wurde ein vereinfachtes Verfahren, Rückabbildung genannt, implementiert, das zu einem gegebenen Expertenbereich eine Obermenge der Datensätze selektiert, die in diesem Expertenbereich enthalten sind. Diese können dann gelesen und in das konzeptionelle Schema abgebildet werden. Anschließend werden sie einzeln getestet, ob sie tatsächlich im Expertenbereich enthalten sind.

Die genaue Wahl der Rückabbildung hat wesentlichen Einfluss auf die Laufzeit der gesamten Prozedur: ist sie zu komplex, dauert ihre Ausführung auf der Datenbank zu lange, ist sie zu einfach, müssen unnötig viele Datensätze gelesen und interpretiert werden.

[15] Freies Zitat nach Herrn Schweier, Bayer AG.

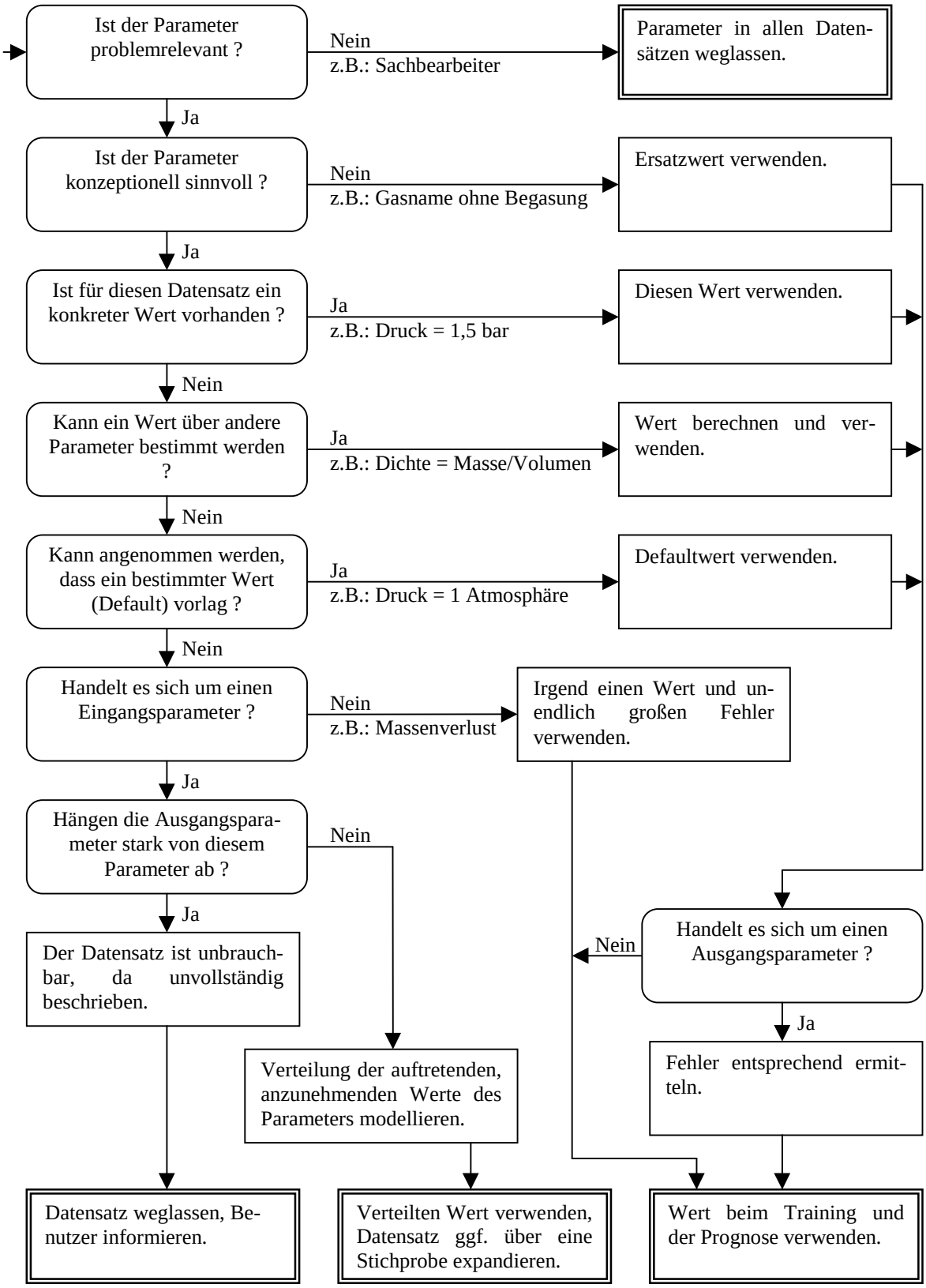

Abbildung 5.5: Vereinfachte Übersicht über die Datenmodellierung.

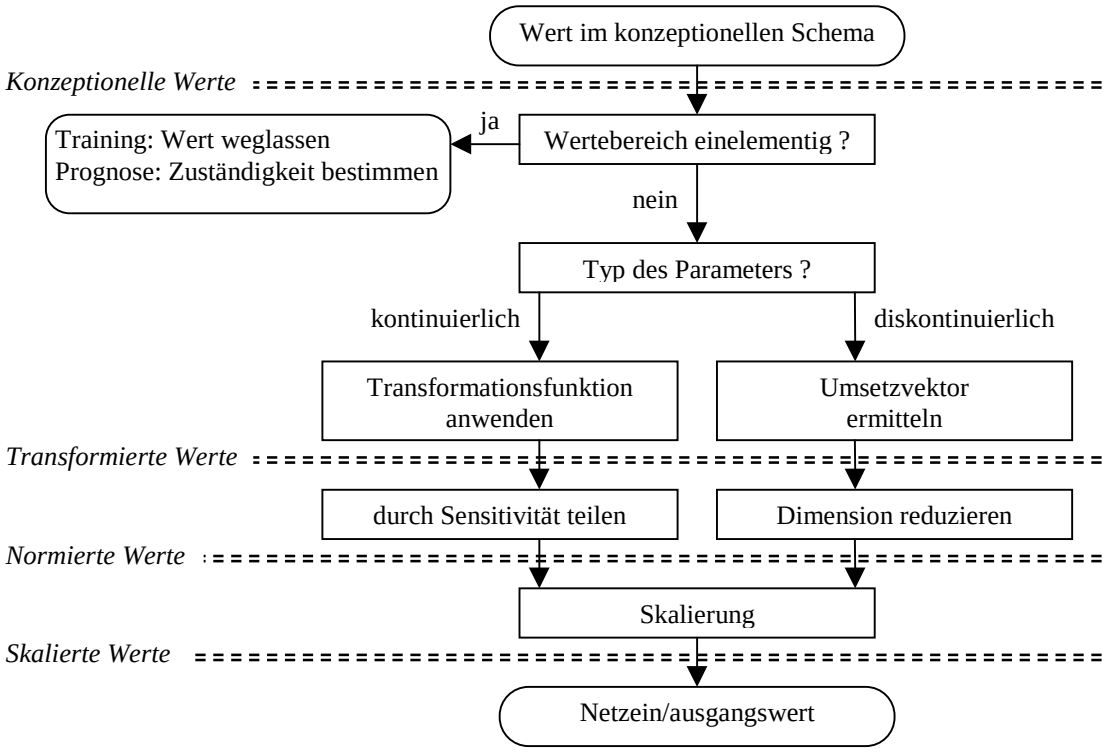

Abbildung 5.6: Übersicht über die Transformation.

5.4 Transformation der Daten

Das Datenmodell der neuronalen Netze ist das einfachste der drei Datenmodelle aus Abbildung 5.1. Ein Netz kommuniziert mit den anderen Teilen des Systems über einen Vektor fester Länge von kontinuierlichen Eingangsvariablen und einen weiteren Vektor fester Länge von kontinuierlichen Ausgangsvariablen. Bei der Verwendung von bayesschen Methoden nach Kapitel 3 besteht jede Ausgangsvariable aus zwei Zahlen, dem Wert und dem zugehörigen Fehler, während die Eingangsvariablen aus nur einer Zahl, ihrem Wert, bestehen.

Im Folgenden sollen nun die Methoden der Abbildung diskutiert werden, die die Daten des konzeptionellen Schemas auf Netztrainings- und -prognosedaten transformiert. Die Transformation der Eingangsparameter auf Netzeingangsvariablen bildet dabei die Vorverarbeitung der Daten, die Berechnung von Trainingswerten und -fehlern für die Netzausgänge sowie die Rücktransformation der Netzprognosen in das konzeptionelle Schema ist die Nachverarbeitung.

Die Transformation besteht aus mehreren Schritten, die in Abbildung 5.6 dargestellt werden. Die ersten (oberen) dieser Schritte sind mehr durch das konzeptionelle Schema bedingt, die späteren (unteren) sind aufgrund bestimmter Eigenschaften der Netze notwendig.

In der Literatur werden einige Verfahren zur automatischen Vorverarbeitung von Rohdaten diskutiert. Dazu zählen die Selektion von Features oder allgemeinere Dimensionsreduktionen des Eingangsraums ([Battiti], [Bidasaria], [BleOba], [CheAnd], [Kulikowski]), oder, im Zusammenhang mit bayesschen Methoden, automatic relevance determination ([BioMeePot], [MacKay4], [PenRob], [Thodberg]). Es ist jedoch günstiger, Problemwissen explizit in die Vor- und Nachverarbeitung einfließen zu lassen, als es experten-lokal aus den Daten zu ermitteln. Denn im zweiten Fall werden einige Informationen, die in den Trainingsdaten enthalten sind, doppelt, also redundant, genutzt, was zu Schwierigkeiten wegen stochastischer Abhängigkeiten führen kann. Umgekehrt ist es dagegen sinnvoll, möglichst viele Informationen über das Problem zu nutzen, zumal sehr wenige Daten im Vergleich zur Dimension des Eingangsraums zur Verfügung stehen.

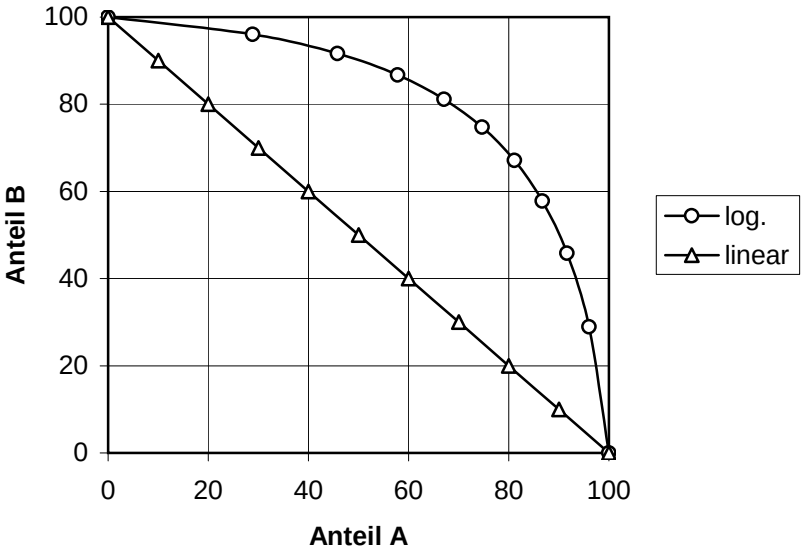

Abbildung 5.7: Abhängigkeiten in den Eingangsparametern. Für ein Medium mit zwei Bestandteilen wurden die Prozentanteile einmal jeweils linear transformiert, $\phi_1(z) = z$, und einmal jeweils logarithmisch (und auf die gleichen Achsen skaliert), $\phi_1(z) = 100(\ln(z + z_0) - \ln(z_0))/(\ln(100 + z_0) - \ln(z_0))$ für hier $z_0 = 10$. Jede durch 10 teilbare Konzentration ist durch ein Symbol dargestellt. Man beachte, dass sich bei der logarithmischen Transformation eine Konzentrationsänderung unterschiedlich auf die euklidischen Abstände der transformierten Werte auswirkt: ist eines der Bestandteile nur als Spur vertreten, ist dieser Abstand relativ groß, sind beide Bestandteile in etwa gleichen Anteilen vorhanden, ist er eher klein. Dies ist eine heuristische Approximation an die Mischungsentropie des Mediums, siehe dazu [BerSch].

5.4.1 Transformation kontinuierlicher Werte

Im Prinzip könnten die Werte eines kontinuierlichen Parameters direkt an einen Netzein- oder -ausgang übernommen werden. Allerdings kann die Generalisierungsfähigkeit eines Experten verbessert werden, wenn die Differenz zweier transformierter Werte der inhaltlichen Ähnlichkeit der Werte bezüglich des Problems entspricht. Die Abbildung, die hier diese inhaltliche Ähnlichkeit auf eine numerische Ähnlichkeit abbildet, wird Transformationsfunktion $\phi : \mathbb{R} \to \mathbb{R}$ genannt; ϕ hängt vom jeweiligen Parameter des konzeptionellen Schemas ab und bildet einen Wert z für diesen Parameter auf den transformierten Wert $\phi(z)$ ab.

Als Beispiel für die Notwendigkeit einer nicht-linearen Funktion ϕ soll hier die Wasserstoffionenaktivität einer Lösung dienen. Diese immer positive Größe schwankt zwischen technisch häufig verwendeten Lösungen um etliche Zehnerpotenzen, wobei die Eigenschaften einer Lösung sehr wesentlich von der konkreten Zehnerpotenz abhängen. Für ein neuronales Netz wäre die direkte Verwendung der Größe Wasserstoffionenaktivität ungünstig, da sich geringe Zehnerpotenzen numerisch kaum unterscheiden, obwohl sie sehr unterschiedliche Eigenschaften der Lösung hervorrufen. Daher wird in der Chemie anstelle der Wasserstoffionenaktivität üblicherweise der pH-Wert[16] verwendet. So ändert sich die Farbe eines Lackmuspapierstreifens in einer Lösung als Funktion der Wasserstoffionenaktivität sehr ungleichmäßig während sie sich als Funktion des pH-Werts deutlich gleichmäßiger ändert und sogar ein gängiges Messverfahren für ihn darstellt.

Alle verwendeten Funktionen sind stetig und streng monoton[17] (und somit eindeutig umkehrbar) und lassen sich sowohl auf Eingangs- als auch auf Ausgangsparameter anwenden.

[16] pH := -log aH+, wobei aH+ die Wasserstoffionenaktivität der Messlösung und pH den pH-Wert bezeichnen.

[17] Prinzipiell könnten auch teilweise konstante Transformationsfunktionen verwendet werden, wenn nämlich die ursprüngliche Größe in bestimmten Bereichen zu gleichem Verhalten bezüglich des Problems führt. Als Beispiel kann etwa die Lösung eines Stoffes in einem Medium dienen, wobei sich die Eigenschaften des Mediums nicht mehr ändern, wenn die Lösung gesättigt ist.

Die Transformationsfunktion ϕ bildet einen kontinuierlichen Wert des konzeptionellen Schemas z ähnlichkeitstreu ab. In der Implementierung wurden folgende Grundtypen verwendet:

- Logarithmische Transformation: $\phi(z) = \ln(z + z_0)$. Dies ist die häufigste Transformationsart der kontinuierlichen Parameter. Lässt der Parameter nur positive Werte zu und schwanken diese um mehrere Zehnerpotenzen, so kann die logarithmische Transformation sinnvoll sein. Ein Beispiel dafür die ist oben genannte Wasserstoffionenaktivität.
  Darf der Parameter auch den Wert 0 annehmen, muss die Konstante z_0 positiv gewählt werden. Sie gibt in etwa an, ab welcher unteren Schwelle eine Unterscheidung der kleinen Werte nicht mehr erfolgen soll. Abbildung 5.7 zeigt ein spezielles Beispiel.
  Formal ist die Anwendung der Logarithmusfunktion auf die Summe $z + z_0$ problematisch, da diese eine Messgröße ist und daher eine physikalische Einheit besitzt. Tatsächlich ist dies aber unkritisch, da ein Wechsel der Einheit lediglich die Addition einer Konstante bewirkt und dies durch die nachfolgende Skalierung ausgeglichen wird (Abbildung 5.6).

- Sigmoide Transformation: $\phi(z) = 1/(1 + \exp(-(z - z_0)/\alpha))$. Dieser Transformationstyp wird angewendet, wenn die Werte eines Parameters nur in einem bestimmten Bereich problemrelevant sind, wenn also bei sehr großen und sehr kleinen Werten der genaue Wert keine Rolle spielt. Mit dem Parameter z_0 wird der Mittelwert des interessanten Bereichs und mit α seine Breite beschrieben.

- Lineare Transformation: $\phi(z) = z$. Parameter, die bereits eine geeignete Ähnlichkeitsdarstellung besitzen, wie beispielsweise der pH-Wert, werden nicht transformiert, d.h. der Wert bleibt unverändert.
  Man beachte, dass die lineare Transformation bis auf einen konstanten Faktor, der aber durch die nachfolgende Division durch die Sensitivität ausgeglichen werden kann, einen Grenzfall der anderen Transformationen darstellt. Dies ist bei der logarithmischen Transformation für $z_0 \to \infty$ und bei der sigmoiden Transformation für $\alpha \to \infty$ der Fall.

- Stückweise Transformation. Die Transformationsfunktion ϕ kann sich stückweise aus den anderen Grundtypen zusammensetzen

$$\phi(z) \quad = \quad \begin{cases} \alpha_1\phi_1(z + z_1), & \text{falls} \quad z < Z \\ \alpha_2\phi_2(z + z_2), & \text{falls} \quad z \geq Z \end{cases} , \tag{5.21}$$

  wobei natürlich auch mehr als zwei Stücke realisierbar sind. Die Konstanten α_1, α_2, z_1 und z_2 sollten so gewählt werden, dass ϕ stetig und monoton ist.
  Ein Beispiel für die Notwendigkeit einer stückweisen Transformation ist die Abtragungsgeschwindigkeit: sie schwankt typischerweise um mehrere Größenordnungen und hat daher logarithmischen Charakter. Allerdings besitzt sie sowohl positive als auch negative Werte. Ihre Transformationsfunktion wurde daher aus einer linearen Funktion für betragsmäßig kleine Werte und zwei logarithmischen Transformationsfunktionen für die positiven bzw. negativen Werte zusammengesetzt.

Die Wahl der Transformationsfunktion ϕ ist nicht immer klar, insbesondere die Festlegung der Parameter z_0 und α ist alles andere als eindeutig. Sicherlich ist es wüschenswert, wenn die Transformationsfunktion wie in der vorliegenden Software a priori für jeden Parameter aufgrund von Wissen über den Parameter und das zugrunde liegende Phänomen festgelegt werden kann. Ist dies nicht möglich, kann man jedoch auch eine automatische Bestimmung der Transformation aufgrund der statistischen Verteilung der Trainingswerte des Parameters vornehmen. Heuristische Ansätze dazu sind in [Möbius] beschrieben.

5.4.2 Parameter-Sensitivität

Wie Abschnitt 3.3.8 nahelegt, sollte sich die Ähnlichkeit zweier Stellen im konzeptionellen Schema im euklidischen Abstand der Netzeingangsvektoren wiederspiegeln. Diese Aussage wurde dort zwar nur für generalisierte lineare Netze diskutiert, jedoch kann man sich leicht überlegen, dass ähnliche Aussagen auch für andere Netztypen wie RBF-Netze mit variablen Zentren oder feed-forward-Netze mit Gewichtsregularisierung gelten.

Das euklidische Abstandsmaß setzt voraus, dass die einzelnen Netzeingänge vergleichbar sind. Dazu müssen sie normiert werden. Für kontinuierliche Werte geschieht diese Normierung mit Hilfe der Sensitivität des konzeptionellen Parameters.

Die Sensitivität c eines Eingangsparameters ist eine Größe, die angibt, wie stark sich ein Wert des Parameters in etwa ändern muss, damit sich bei der wahren Funktion eine signifikante Änderung der Ausgangsgrößen ergibt. Man sieht, dass diese Definition äußerst vage und eher intuitiv ist. Bei der Festlegung der Sensitivitäten einzelner Parameter geht es aber auch nur um die Abschätzung der richtigen Größenordnung und der grob richtigen Verhältnisse zwischen den Parametern.

Die Angabe der Sensitivität hängt natürlich von der Transformationsfunktion ab. Ist diese linear, dann hat die Konstante c die gleiche physikalische Einheit wie die Werte des Parameters, ist sie logarithmisch, dann wird die Sensitivität entsprechend logarithmisch angegeben. Betrachten wir dazu beispielhaft zwei Parameter:

Parameter	pH-Wert	Druck
Transformationstyp	linear	logarithmisch mit Konstante $z_0 = 0,1\text{bar}$
Sensitivität	1pH	Faktor 2
konzeptioneller Wert	z_1	z_2
transformierter Wert	z_1	$\ln(z_2 + 0,1\text{bar})$
normierter Wert	$\frac{z_1}{1\text{pH}}$	$\frac{\ln(z_2 + 0,1\text{bar})}{\ln(2)}$

Die Wahl genau dieser Sensitivitäten besagt hier, dass sich eine Erhöhung des pH-Werts um 1pH ähnlich stark auf die Korrosionseigenschaften auswirken kann wie eine Verdopplung oder Halbierung des Drucks (für Drücke größer als z_0).

Bei den Sensitivitäten handelt es sich also um a priori Größen, die heuristisch durch die Modellierung der Daten vorgegeben sind. Die tatsächlichen (a posteriori) Wirkungen von Änderungen in einem Eingangsparameter auf das Korrosionsverhalten sind natürlich vom gesamten Korrosionssystem abhängig: natürlich von der Stelle, also allen Eingangsparametern, aber auch von den Ausgangsparametern, denn während sich vielleicht die Abtragungsgeschwindigkeit kaum ändert wechselt aber die Lochfraßanfälligkeit. Mehr zur Sensitivität findet sich in Abschnitt 5.4.6.

5.4.3 Umsetzung diskontinuierlicher Ausprägungen

Die Werte diskontinuierlicher Eingangsparameter sind Ausprägungen, die zueinander in bestimmten Ähnlichkeiten stehen. Dabei unterscheidet man zwei verschiedene Typen: bekannte und unbekannte Ähnlichkeitsmaße. Bei den meisten Parametern ist die Menge der Ausprägungen und ihre Semantik bekannt, sodass ein Ähnlichkeitsmaß in Form einer symmetrischen Matrix mit verschwindender Diagonale angegeben werden kann. Ist die Menge der Ausprägungen dagegen unbekannt, weil sie etwa dynamisch wachsen kann, so kann kein detailliertes Ähnlichkeitsmaß angegeben werden. Man verwendet in diesem Fall in der Regel das Abstandsmaß des Kronecker-Deltas: zwei verschiedene Ausprägungen haben immer den Abstand 1.

Man beachte, dass die Ähnlichkeiten der Ausprägungen nicht nur untereinander stimmig sein sollten sondern auch mit den Ähnlichkeiten anderer Parameter korrespondieren sollten. In dem Ähnlichkeitsmaß ist also die Sensitivität des Parameters ähnlich der Sensitivität eines kontinuierlichen Parameters enthalten.

Um eine ähnlichkeitstreue Abbildung eines diskontinuierlichen Parameters auf die Netzeingänge zu realisieren, müssen für diesen einen Parameter in der Regel mehrere Netzeingänge erzeugt werden. Daher wird für jede Ausprägung a ein sogenannter Umsetzvektor $v(a)$ definiert, der die Werte der Netzeingänge enthält. Alle Umsetzvektoren haben dabei natürlich die gleiche Dimension, die hier mit k bezeichnet werden soll. Hier ein Beispiel für die Umsetzvektoren eines Parameters “Farbe“, wenn das zugrunde liegende Phänomen auf additiver Farbmischung nach dem RGB-Schema basiert:

Ausprägung a	$v_1(a)$	$v_2(a)$	$v_3(a)$
rot	1	0	0
grün	0	1	0
blau	0	0	1
gelb	1	1	0
grau	0,5	0,5	0,5

Wenn das Ähnlichkeitsmaß die Axiome einer Metrik erfüllt, dann können immer Umsetzvektoren gefunden werden, die die Ausprägungen bezüglich des euklidischen Abstands im Netzeingangsraum exakt ähnlichkeitstreu abbilden. Bei j Ausprägungen werden dabei höchstens $j-1$ Netzeingänge benötigt.

In der Praxis ist die Einhaltung der Dreiecksungleichung bei der Festlegung der Ähnlichkeit nicht einfach, da möglicherweise sehr viele Kombinationen geprüft werden müssen. Es ist daher oft einfacher, die Umsetzvektoren direkt festzulegen. Dabei kann auch eine höhere Dimension als mathematisch notwendig in Kauf genommen werden kann, denn wie unten beschrieben kann die Anzahl der Netzeingänge automatisch minimiert werden. Bei dynamischen Parametern wird der Einfachheit halber eine 1-aus-k-Kodierung verwendet, die offensichtlich eine Dimension mehr als notwendig definiert.

Die Minimierung der Dimension ist sinnvoll, um die Anzahl der Netzeingänge klein zu halten, wenn dabei keine Information verloren geht. Sie eliminiert auch affin lineare Abhängigkeiten in den Netzeingängen, die unter Umständen zu numerischen Problemen führen können, siehe dazu die Abschnitte 3.2.3 und 3.3.3. Wir gehen daher im Folgenden davon aus, dass die Ähnlichkeiten der Werte des Parameters die einzige Information darstellen. Die automatische Dimensionsminimierung ist notwendig, wenn nicht alle Ausprägungen in den Trainingsdaten vorkommen. Sind beispielsweise beim Parameter „Farbe“ nur die Ausprägungen *rot* und *grün* in den Trainingsdaten vorhanden, so kann die Umsetzkomponente v_3 ganz offensichtlich weggelassen werden, denn sie ist konstant. Die Komponenten v_1 und v_2 sind außerdem redundant, denn sie verbindet für diese Trainingsdaten die affin lineare Abhängigkeit $v_1 + v_2 = 1$. Eine mögliche, dimensions-minimale Umsetzung wäre also $rot \mapsto (0)$ und $grün \mapsto (\sqrt{2})$.

Die Dimensionsreduktion soll nun allgemein hergeleitet und ein Algorithmus konstruiert werden. Seien $v_1, \ldots, v_J \in \mathbb{R}^k$ die Umsetzvektoren der Ausprägungen, die in den aktuellen Trainingsdaten vorkommen. Gesucht werden nun Vektoren $\tilde{v}_1, \ldots, \tilde{v}_J \in \mathbb{R}^{\tilde{k}}$, die die gleichen euklidischen Abstände,

$$\forall i, j \in \{1, \ldots, J\} : \ \|\tilde{v}_i - \tilde{v}_j\| = \|v_i - v_j\|, \tag{5.22}$$

und minimale Dimension $\tilde{k}$ besitzen.

Eine Translation im Eingangsraum der Netze ist aufgrund der nachfolgenden Skalierung (Abschnitt 5.4.5) eine Invariante. Daher wird zunächst der Vektor v_J auf den Ursprung verschoben. Sein reduzierter Vektor $\tilde{v_J}$ ist dann ebenfalls der Nullvektor und alle übrigen reduzierten Vektoren lassen sich durch eine lineare Abbildung aus den verschobenen bestimmen. Sei daher die Matrix R spaltenweise durch die übrigen verschobenen Umsetzvektoren gegeben:

$$R \ := \ \Big((v_1 - v_J) \cdots (v_{J-1} - v_J) \Big) \ \in \mathbb{R}^{k \times (J-1)}. \tag{5.23}$$

Der j-te Umsetzvektor kann nun durch den Ausdruck

$$v_j \ = \ R u_j + v_J \tag{5.24}$$

mit

$$\begin{aligned} u_j \ &\in \ \mathbb{R}^{J-1} \\ u_j \ &= \ \begin{cases} \text{0-Vektor} & \text{falls } j = J \\ j\text{-ter Einheitsvektor} & \text{sonst} \end{cases} \end{aligned} \tag{5.25}$$

dargestellt werden.

Für die Matrix R wird nun eine Singulärwertzerlegung $R = UDV^T$ berechnet (ein Algorithmus dazu findet sich z.B. in [PreTeuVet]). Dabei ist U eine Orthogonalmatrix mit $U^T U = I$, D eine Diagonalmatrix mit nicht-negativen und absteigend sortierten Diagonalelementen und V eine Orthogonalmatrix mit

$V^TV = I$. Die Dimensionalitäten der Matrizen bestimmen sich aufgrund einer Fallunterscheidung:

$$\begin{array}{llll} U \in \mathbb{R}^{k\times(J-1)} & D \in \mathbb{R}^{(J-1)\times(J-1)} & V \in \mathbb{R}^{(J-1)\times(J-1)} & \text{falls } J-1 < k \\ U \in \mathbb{R}^{k\times k} & D \in \mathbb{R}^{k\times k} & V \in \mathbb{R}^{(J-1)\times k} & \text{falls } J-1 \geq k. \end{array} \tag{5.26}$$

Im Algorithmus muss diese Fallunterscheidung nachvollzogen werden.

Man kann nun die Dimension der Matrix D reduzieren, indem man rechts unten verschwindende Elemente weglässt und dann deren Zeilen und Spalten eliminiert. Aufgrund der Numerik ist es dabei ausreichend einen kleinen Schwellwert $\epsilon > 0$ festzulegen und alle Elemente kleiner ϵ wegzulassen. Der Wert von ϵ sollte dabei kleiner als der kleinste Abstand zweier Umsetzvektoren sein. Zu der reduzierten Matrix $\tilde{D} \in \mathbb{R}^{\tilde{k}\times\tilde{k}}$ gehören dann auch entsprechend durch Spaltenelimination reduzierte Matrizen $\tilde{U}$ und $\tilde{V}$, sodass die Zerlegung $R = \tilde{U}\tilde{D}\tilde{V}^T$ erhalten bleibt. Es gilt

$$\tilde{U} \in \mathbb{R}^{k\times\tilde{k}} \quad \tilde{D} \in \mathbb{R}^{\tilde{k}\times\tilde{k}} \quad \tilde{V} \in \mathbb{R}^{(J-1)\times\tilde{k}}. \tag{5.27}$$

Man berechnet nun die reduzierten Umsetzvektoren als $\tilde{v_j} := \tilde{D}\tilde{V}^T u_j$ und Bedingung 5.22 wird erfüllt, denn für jedes $i, j \in \{1, \ldots, J\}$ gilt:

$$\begin{aligned} \|\tilde{v_i} - \tilde{v_j}\| &= \|\tilde{D}\tilde{V}^T(u_i - u_j)\| \\ &= (u_i - u_j)^T\tilde{V}\tilde{D}\tilde{D}\tilde{V}^T(u_i - u_j) \\ &= (u_i - u_j)^T\tilde{V}\tilde{D}\tilde{U}^T\tilde{U}\tilde{D}\tilde{V}^T(u_i - u_j) \\ &= (u_i - u_j)^T R^T R(u_i - u_j) \\ &= \|R(u_i - u_j)\| \\ &= \|Ru_i + v_J - Ru_j - v_J\| \\ &= \|v_i - v_j\|. \end{aligned} \tag{5.28}$$

Aus dem beschriebenen Verfahren zur Dimensionsminimierung lässt sich direkt ein Algorithmus ableiten.

Das Verfahren minimiert zwar die Anzahl der Netzeingänge, jedoch sind prinzipiell viele Lösungen der Umsetzvektoren $\tilde{v_1}, \ldots \tilde{v_J} \in \mathbb{R}^{\tilde{k}}$, die die Bedingung 5.22 erfüllen, möglich. Es ist daher zu überlegen, ob es weitere sinnvolle Forderungen gibt, die das Netztraining und/oder die Generalisierungsfähigkeit verbessern. Ein Möglichkeit dazu wäre die Minimierung des durch die Umsetzvektoren aufgespannten Lernraums. Dieser hängt natürlich von der Netzfunktion ab, in der gewählten Implementierung ist es der kleinste achsenparallele Quader, der alle Trainingsstellen enthält. Es besteht hier also noch Forschungsbedarf.

5.4.4 Ersatzwerte

Nicht jeder Parameter existiert in jedem Korrosionssystem. So kann es passieren, dass es in einer Trainingsdatenmenge einen abhängigen Parameter gibt, der für einige Datensätze existiert und für andere nicht. Hat dieser Parameter in den Datensätzen, in denen er existiert, mehr als nur einen Wert, so erzeugt er mindestens einen Netzeingang. Dieser muss aber auch mit einem Wert belegt werden, damit das Training bzw. die Prognose durchgeführt werden können. Dieser Wert ist nun der sogenannte Ersatzwert.

Der Ersatzwert wird für jeden abhängigen Parameter festgelegt und sollte nach Möglichkeit den inhaltlichen Grenzfall zur Nichtexistenz des Parameters beschreiben. Der Parameter „Gasmenge“, der nur im Falle der Begasung existiert, hat den Ersatzwert 0, der am ehesten die Nicht-Begasung beschreibt.

Der Ersatzwert ist bei kontinuierlichen Parametern eine Größe in der entsprechenden physikalischen Einheit des Parameters, die noch transformiert werden muss. Dabei muss er nicht notwendigerweise aus dem Wertebereich des Parameters gewählt werden: während echte „Gasmengen“ immer positiv sein müssen, darf der Ersatzwert durchaus den Wert 0 annehmen.

Bei diskontinuierlichen Parametern wird der Ersatzwert durch einen Umsetzvektor definiert, der nicht notwendigerweise dem Umsetzvektor einer Ausprägung gleichen muss. Dadurch wird vermieden, dass eine neue Ausprägung geschaffen werden muss, um den Ersatzwert sinnvoll auf die Netzeingänge abbilden zu können.

5.4.5 Skalierung

Die Skalierung der Netzdaten ist die letzte Verarbeitungsstufe der Transformation und dient dazu, die Daten auf die spezielle Netzarchitektur vorzubereiten. Die Skalierung hängt daher direkt vom verwendeten Netztyp ab.

Beispiele für eine notwendige Skalierung sind etwa Feed-forward-Netze mit sigmoiden Ausgangsneuronen. Diese liefern beschränkte Prognosen, bei der logistischen Aktivierungsfunktion etwa Prognosen aus dem Intervall $]0,1[$. Daher sollten die Trainingswerte der Netzausgänge ebenfalls in diesem Intervall liegen, was etwa durch eine affin lineare Skalierung realisiert werden kann.

Im Fall der verwendeten generalisierten linearen Netze ist vor allem eine Skalierung der Eingänge nötig. Wie in Abschnitt 3.4.1 beschrieben, erwartet das Netz, dass die Trainingsstellen in einem achsenparallelen Quader liegen. Die Werte des l-ten Eingangs liegen dabei nach Gleichung 3.124 in einem symmetrischen Intervall um 0 mit vorgegebener Intervallbreite $2\xi_l$. Diese Forderung ist leicht durch eine Translation zu erfüllen: sind $x_{1l}, \ldots, x_{Nl}$ die normierten Trainingswerte des l-ten Eingangs, dann sind $x_{1l}+\delta, \ldots, x_{Nl}+\delta$ die skalierten Trainingswerte mit

$$\delta = -\frac{1}{2}\left(\min\{x_{1l}, \ldots, x_{Nl}\} + \max\{x_{1l}, \ldots, x_{Nl}\}\right) \tag{5.29}$$

$$\xi_l = \frac{1}{2}\left(\max\{x_{1l}, \ldots, x_{Nl}\} - \min\{x_{1l}, \ldots, x_{Nl}\}\right). \tag{5.30}$$

Man beachte, dass die Translation im Netzeingangsraum keine eigentliche Wirkung auf die Prognosen hat, sondern nur die Konvention zur Platzierung der Basisfunktionen erfüllt.

5.4.6 Singuläre Trainingsdaten

Nehmen wir an, dass alle Trainingsdaten eines Experten in einem bestimmten Parameter den gleichen Wert besitzen. Dies ist in der Praxis bei vielen Parametern der Fall, beispielsweise wenn keiner der Werkstoffe in den Trainingsdaten bestimmte Sonderlegierungselemente enthält: der Parameter „Gold-Massenprozent“ hat dann für alle Trainingsdatensätze den Wert 0.

In einem System kooperierender Netze muss auch dieses Netz eine Prognose berechnen können, wenn der Anfragewert nicht exakt mit dem trainierten Wert übereinstimmt, da sonst die Prognose der Kooperation unstetig wäre. Ein Werkstoff ändert seine Eigenschaften schließlich nicht fundamental, wenn ihm eine kleine Spur Gold hinzugefügt wird. Allerdings sollte der Prognosefehler umso größer werden, je weiter der Anfragewert und der trainierte Wert voneinander abweichen.

Wie stark der Prognosefehler ansteigen soll, kann das Netz nicht berechnen, weil es natürlich keine Vergleichsmöglichkeit in den Trainingsdaten besitzt. Daher muss die Vergrößerung des Prognosefehlers durch die Transformation erfolgen, weil nur hier spezielles a priori Wissen über das zugrunde liegende Phänomen einfließen kann.

Der Begriff der Sensitivität wird hier für diesen Zweck formalisiert. Betrachten wir einen festen Eingangsparameter e und einen festen Ausgangsparameter a. Sei x_e ein transformierter Wert von e, x_a ein transformierter Wert von a, x der Vektor aller Eingangswerte inklusive x_e und f_a die wahre Funktion für a in transformierten Werten, dann gilt

$$\begin{aligned} x_a &= f_a(x) \\ &= f_a(x_1, \ldots, x_e, \ldots, x_L). \end{aligned} \tag{5.31}$$

Die Abhängigkeit des Ausgangs a vom Eingang e an der Stelle x ist nun durch die partielle Ableitung $\partial x_a / \partial x_e$ gegeben. Über diese werden nun folgende Annahmen gemacht:

$$E\left[\frac{\partial f_a}{\partial x_e}(x)\right] = 0 \tag{5.32}$$

$$VAR\left[\frac{\partial f_a}{\partial x_e}(x)\right] = c_{ea}^2, \tag{5.33}$$

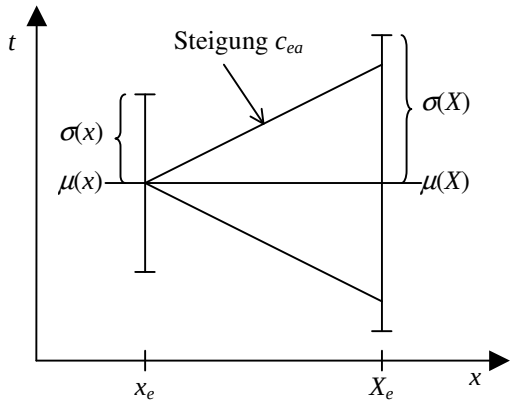

Abbildung 5.8: Prognosefehler bei Prognosen neben dem trainierten Wert.

wobei c_{ea} die Sensitivität[18] von e bezüglich a ist. Die Erwartungswerte werden über alle Stellen x gebildet. Gleichung 5.32 drückt aus, dass kein a priori Wissen über die Tendenz oder das Vorzeichen der Abhängigkeit zwischen e und a existiert, Gleichung 5.33 verbindet die Größenordnung der Abhängigkeit mit der Sensitivität, die hier auch auf den Ausgangsparameter bezogen ist.

Sei nun x_e der transformierte Wert von e in den Trainingsdaten und X_e der transformierte Wert, bei dem eine Prognose berechnet werden soll. Der Parameter e bildet natürlich keinen Netzeingang, daher wird zunächst eine Prognose des Netzes an der semantischen Stelle x_e berechnet. Das Ergebnis ist eine Zufallsvariable $t|x_e$ nach Ausdruck 3.12, die $\mathcal{N}\left(\mu_a(x), \sigma_a^2(x)\right)$-verteilt ist.

Abbildung 5.8 zeigt nun das Prinzip der Vergrößerung des Prognosefehlers. Unter linearer Approximation der wahren Funktion f_a bezüglich x_e gilt nun für die Variable t an der eigentlichen Prognosestelle

$$t|X_e \quad = \quad t|x_e + (X_e - x_e) \cdot \frac{\partial f_a}{\partial x_e}(x). \tag{5.34}$$

Da $t|x_e$ nicht von e und der Differenzialquotient a priori nur von e abhängt, sind diese beiden Zufallsvariablen stochastisch unabhängig und es folgt

$$\begin{aligned}
\mu_a(X) &= E[t|X_e] \\
&= E[t|x_e] + (X_e - x_e) \cdot E\left[\frac{\partial f_a}{\partial x_e}(x)\right] \\
&= \mu_a(x) \quad \text{und} \qquad (5.35) \\
\sigma_a^2(X) &= VAR[t|X_e] \\
&= VAR[t|x_e] + (X_e - x_e)^2 \cdot VAR\left[\frac{\partial f_a}{\partial x_e}(x)\right] \\
&= \sigma_a^2(x) + (X_e - x_e)^2 c_{ea}^2. \qquad (5.36)
\end{aligned}$$

Das Verhalten ist wie erwartet: während der Prognosewert unverändert bleibt, wird der Prognosefehler mit zunehmendem Abstand zwischen trainiertem und angefragtem Wert größer. Bei mehreren Parametern, die in den Trainingsdaten nur einen einzigen Wert aufweisen, gilt dann entsprechend für den Prognosefehler

$$\sigma_a(X) \quad = \quad \sqrt{\sigma_a^2(x) + \sum_{e:\ e \text{ hat singuläre Trainingsdaten}} (X_e - x_e)^2 c_{ea}^2}. \tag{5.37}$$

In der Praxis möchte man natürlich nicht für jede Kombination aus Eingangsparameter e und Ausgangsparameter a eine Sensitivität festlegen. Man kann daher für jeden Parameter eine eigene Sensitivität

[18] Diese ist nicht gleich der Sensitivität aus Abschnitt 5.4.2, s.u..

festlegen, c_e und c_a, und für die gemeinsame gilt dann $c_{ea} = c_a/c_e$ entsprechend dem Differenzialquotienten aus Gleichung 5.33. Diese Sensitivitäten entsprechen dann wieder denen aus Abschnitt 5.4.2.

5.4.7 Verteilte Werte

Soll ein Experte auf einer bestimmten Menge von Datensätzen trainiert werden, so trifft für jeden Parameter, der eine Defaultverteilung nach Abschnitt 5.2.4 zulässt, genau einer der folgenden Fälle zu:

1. Alle Datensätze haben konkrete Werte für diesen Parameter. Dies ist der einfachste Fall, und der Parameter wird wie alle anderen Parameter auch behandelt.

2. Alle Datensätze haben einen verteilten Wert für diesen Parameter. Hier macht man sich zu Nutze, dass alle Werte identisch verteilt sind: der Parameter wird nicht zu einem Eingangsparameter des Netzes und die Verteilung wird erst zum Prognosezeitpunkt berücksichtigt. Bei einer Prognose wird dann der Prognosefehler vergrößert, da der angefragte Wert nur mit einer gewissen Wahrscheinlichkeit mit dem tatsächlich bei der Messung vorgelegenen Wert übereinstimmt.
   Sei e ein kontinuierlicher Eingangsparameter mit Defaultverteilung, a ein Ausgangsparameter, X_e ein transformierter Wert von e, an dem eine Prognose berechnet werden soll, x_e die transformierte Zufallsvariable der Defaultverteilung von e, x der Vektor aller Netzeingangsvariablen, t die Netzausgangsvariable und f_a die wahre Funktion, dann gilt auch hier Gleichung 5.34 entsprechend:

$$t|X_e \quad = \quad t|x_e + (X_e - x_e) \cdot \frac{\partial f_a}{\partial x_e}(x). \tag{5.38}$$

   In dieser Gleichung gibt es nun drei Zufallsvariablen, die stochastisch unabhängig sind: erstens die Netzprognose $t|x_e \propto \mathcal{N}\left(\mu_a(x), \sigma_a^2(x)\right)$, zweitens die Defaultverteilung, die in der praktischen Implementierung eine Normalverteilung $x_e \propto \mathcal{N}\left(\mu_D, \sigma_D^2\right)$ (im transformierten Zustand) ist, und drittens der Differenzialquotient, der hier ebenfalls als normalverteilt $\partial f_a/\partial x_e(x) \propto \mathcal{N}\left(0, c_{ea}^2\right)$ und als unabhängig von der Stelle x angenommen wird. Dann folgt für die Prognosen unter Berücksichtigung der Defaultverteilung:

$$\begin{aligned}
\mu_a(X) &= E[t|X_e] \\
&= \mu_a(x) \qquad \text{und} \qquad (5.39) \\
\sigma_a^2(X) &= VAR[t|X_e] \\
&= \sigma_a^2(x) + E\left[(X_e - x_e)^2 \cdot \left(\frac{\partial f_a}{\partial x_e}(x)\right)^2\right] \\
&= \sigma_a^2(x) + E\left[(X_e - x_e)^2\right] \cdot E\left[\left(\frac{\partial f_a}{\partial x_e}(x)\right)^2\right] \\
&= \sigma_a^2(x) + \left((X_e - \mu_D)^2 + \sigma_D^2\right) c_{ea}^2. \qquad (5.40)
\end{aligned}$$

   Es ist nicht verwunderlich, dass die Form der Vergrößerung des Prognosefehlers derer bei singulären Trainingsdaten ähnelt, und in der Tat stellt die Defaultverteilung für $\mu_D \to x_e$ und $\sigma_D \to 0$ den Grenzfall zu singulären Daten dar (Gleichung 5.36).
   Ist der Parameter diskontinuierlich, wird Gleichung 5.38 durch

$$t|X_e \quad = \quad t|x_e + \|v(X_e) - v(x_e)\| \cdot \frac{\partial f_a}{\partial x_e}(x) \tag{5.41}$$

   ersetzt, wobei $v(X_e)$ und $v(x_e)$ nun die Umsetzvektoren der Ausprägungen X_e und x_e sind. Da die Sensitivität bereits in den Umsetzvektoren enthalten ist, wird $\partial f_a/\partial x_e(x) \propto \mathcal{N}(0,1)$ verteilt angenommen. Seien $a_1, \ldots, a_J$ die Ausprägungen von e und $P(a_1), \ldots, P(a_J)$ die Wahrscheinlichkeiten dieser Ausprägungen in der Defaultverteilung, dann ergibt sich für die Gesamtprognosen

$$\mu_a(X) \quad = \quad \mu_a(x) \qquad \text{und} \tag{5.42}$$

$$\begin{aligned}\sigma_a^2(X) &= \sigma_a^2(x) + E\left[\|v(X_e) - v(x_e)\|^2 \cdot \left(\frac{\partial f_a}{\partial x_e}(x)\right)^2\right] \\ &= \sigma_a^2(x) + E\left[\|v(X_e) - v(x_e)\|^2\right] \cdot 1 \\ &= \sigma_a^2(x) + \sum_{j=1}^{J} P(a_j)\|v(X_e) - v(a_j)\|^2. \end{aligned} \tag{5.43}$$

3. Einige Datensätze haben konkrete, einige verteilte Werte für diesen Parameter. Beim Training werden nun alle Datensätze, die einen verteilten Wert besitzen, expandiert. Dabei wird zunächst eine repräsentative Stichprobe aus der Defaultverteilung ermittelt, die für jeden Parameter zusammen mit der Defaultverteilung festgelegt ist. Sei hier $\xi_1, \ldots, \xi_K$ die Stichprobe mit den zugehörigen Wahrscheinlichkeiten $P_1, \ldots, P_K$. Aus jedem Datensatz mit verteiltem Wert und jedem der K Elemente der Stichprobe wird nun ein neuer Datensatz gebildet, indem

   - der verteilte Wert des Parameters durch jeden Wert x_j der Stichprobe ersetzt wird,
   - alle übrigen Ein- und Ausgangsparameter kopiert werden und
   - der Trainingsfehler durch den Faktor $\sqrt{P_j}$ dividiert wird.

   Die Expansion wird noch vor der Transformation durchgeführt, die Werte der Stichprobe werden daher wie alle anderen Werte auch anschließend auf die Netzeingänge abgebildet.
   Die Division (Vergrößerung) des Trainingsfehlers ergibt sich aus folgender Überlegung: die Information (Entropie) eines Datensatzes ist durch ihren Trainingsfehler bestimmt. Für diesen gilt aber die Äquivalenz zwischen einem und mehreren Datensätzen nach Gleichung 3.59 auf Seite 43. Ist s der ursprüngliche Messfehler, dann könnten die expandierten Datensätze mit den Messfehlern $s_1, \ldots, s_K$ nach Gleichung 3.59 zusammengefasst werden, wenn ihre Messstellen gleich wären:

$$\begin{aligned}\left(\sum_{k=1}^{K} s_k^{-2}\right)^{-\frac{1}{2}} &= \left(\sum_{k=1}^{K}\left(\frac{s}{\sqrt{P_j}}\right)^{-2}\right)^{-\frac{1}{2}} \\ &= s\left(\sum_{k=1}^{K} P_j\right)^{-\frac{1}{2}} \\ &= s. \end{aligned} \tag{5.44}$$

   Mit anderen Worten: der ursprüngliche Datensatz enthält genauso viel Information wie die expandierten Datensätze zusammen. Würde man den Messfehler unverändert lassen, dann würde ein Datensatz mit verteiltem Wert beim Training stärker berücksichtigt als ein Datensatz mit konkretem Wert, was wenig plausibel ist.
   Da Gleichung 3.59 formal gleiche Messstellen voraussetzt, ist die beschriebene Festlegung des expandierten Messfehlers ein Modell und keine zwingende Herleitung.

Die Fallunterscheidung und die jeweilige Behandlung der verteilten Werte wurde so gewählt, dass sie zu möglichst ähnlichen Prognosen bei verschiedenen Einteilungen der Trainingsdaten auf Experten führt. Konkret bedeutet dies folgendes: nehmen wir eine feste Menge von Trainingsdaten an, von denen etwa die Hälfte einen konkreten Wert und die andere Hälfte einen verteilten Wert für einen bestimmten Parameter hat. Dann soll eine Kooperation aus zwei Experten, von denen der erste auf allen Datensätzen mit konkreten Werten nach Fall 1 und der zweite auf allen Datensätzen mit verteilten Werten nach Fall 2 trainiert wurde, ähnlich gute Prognosen berechnen wie ein einzelner Experte, der auf allen diesen Trainingsdaten nach Fall 3 trainiert wurde. Der Begriff „ähnlich gut“ ist dabei im Sinne von Abschnitt 4.2 gemeint.

Man beachte, dass im Falle der Kooperation die beiden Experten unabhängig voneinander konstruierbar sein müssen und daher keine a posteriori Sensitivität des Parameters bekannt ist. Somit ist hier viel a priori Information nötig, um die Invarianz bezüglich verschiedener Einteilungen der Trainingsdaten

zu ermöglichen: die Parametersensitivität, die Defaultverteilung (einschließlich der Wahl der Stichprobe) und das Modell zur Bestimmung des expandierten Messfehlers.

Wenn es mehrere Parameter gibt, die eine Defaultverteilung besitzen, kann die Expansion nach Fall 3 zu extrem vielen Datensätzen führen. Dabei gibt es zwei grundsätzliche Vorgehensweisen:

- Expandiert man jeden Parameter für sich, wächst die Anzahl der zu trainierenden Datensätze exponentiell mit der Anzahl dieser Parameter. Man erhält dadurch zwar eine gute Abdeckung des Eingangsraums, jedoch kann das Training leicht inakzeptabel lang dauern.

- Wählt man durch stochastisches Sampling aus der Gesamtverteilung aller zu expandierenden Parameter eine konstante Anzahl von Stichproben aus, so ist zwar die Anzahl der Trainingsdaten beschränkt, es kann aber passieren, dass die Menge der Stichproben den Raum bezüglich des Problems schlecht beschreibt. Außerdem können die Stichproben nicht mehr fest und damit repräsentativ für jeden Parameter festgelegt werden.

In der Implementierung wurden nur insgesamt sieben Parameter definiert, die verteilte Werte zulassen. Die fünf kontinuierlichen von ihnen besitzen je drei und die beiden diskontinuierlichen nur je zwei Stützstellen. Nur ein Parameter existiert in jedem Korrosionssystem, alle anderen sind abhängig und existieren jeder für sich in nur sehr wenigen Trainingsdaten. Daher ist hier nicht mit einer explodierenden Anzahl an Trainingsdaten zu rechnen. Trotzdem gilt folgende Empfehlung für die Einteilung der Daten auf Experten: wenn ein Experte sowohl konkrete als auch verteilte Werte in einem Parameter besitzt und insgesamt mehr als etwa 200 Trainingsdatensätze hat, sollte er so in zwei Experten aufgeteilt werden, dass einer die konkreten und der andere die verteilten Werte erhält.

In der Literatur werden andere Modelle mit dem Umgang fehlender Werte beschrieben. [KatKat] modelliert fehlende Ausgangswerte ebenfalls als Verteilungen und leitet daraus entsprechende Fehlerfunktionen beim Training ab; das Verfahren basiert zwar auf klassischen Netzen, läßt sich jedoch auch auf bayessche Methoden übertragen. In [IshMiyTan] werden fehlende Eingangswerte durch Intervalle abgeschätzt. Diese Intervalle werden dann durch das Netz propagiert und man erhält so entsprechende Intervalle an den Netzausgängen. Leider ist dieses Modell nicht mit den bayesschen Methoden vereinbar.

5.4.8 Expertenzuständigkeit

Das implementierte System basiert auf Experten, die nach Abschnitt 4.1 miteinander kooperieren, um eine Gesamtprognose zu berechnen. Allerdings wird aus Effizienzgründen nicht von jedem Netz eine Einzelprognose berechnet, sondern nur von denjenigen, deren Expertenbereich passend zur Prognosestelle liegt.

In die Kooperation gehen die einzelnen Expertenprognosen umso stärker ein, je kleiner ihr Prognosefehler ist (Gleichungen 4.17 und 4.18). Experten mit sehr großen Prognosefehlern können daher in guter Näherung weggelassen werden, da so Rechenzeit gespart werden kann. Ein Experte ist zuständig für eine gegebene Prognosestelle, wenn nicht bereits durch den Expertenbereich offensichtlich ist, dass sein Prognosefehler dort groß ist.

Liegt die Prognosestelle innerhalb des Expertenbereichs (wobei die Wertebereiche einzelner Parameter um die wahrscheinlichen Werte der Defaultverteilung erweitert werden, wenn der Expertenbereich den verteilten Wert zulässt), ist der Parameter immer zuständig. Natürlich kann es auch innerhalb des Expertenbereichs Stellen mit sehr großen Prognosefehlern geben, wenn diese abseits der Trainingsdaten liegen. Dies kann aber in der Praxis nur durch das Netz selbst erkannt werden. Im Folgenden geht es daher um die Schätzung des Prognosefehlers außerhalb des Expertenbereichs, also um die Extrapolationsfähigkeit eines Experten.

Die praktisch implementierte Lösung ist recht einfach. Kontinuierliche Parameter werden dabei überhaupt nicht beachtet. Um dies zu begründen, müssen zwei Fälle unterschieden werden: spannt der kontinuierliche Eingangsparameter ein Intervall im Expertenbereich auf, so gibt es einen Netzeingang zum Parameter und der Prognosefehler hängt kompliziert von den Basisfunktionen, der Gewichtsregularisierung und den Trainingsdaten ab, eine robuste Schätzung oder untere Schranke ist nicht bekannt. Besitzt der Eingangsparameter nur einen einzelnen Wert im Expertenbereich, so kann zwar prinzipiell der Prognosefehler nach Gleichung 5.36 bzw. bei verteilten Werten nach Gleichung 5.40 geschätzt werden. Allerdings

wird in der Praxis fast immer genau dieser eine Wert bzw. ein wahrscheinlicher Wert im Falle der trainierten Defaultverteilung angefragt, da es sich um einen inhaltlichen Default handelt. Der Aufwand lohnt daher nicht.

Bei den diskontinuierlichen Parametern dagegen ist ein Experte nur zuständig, wenn die angefragte Ausprägung in der Menge der trainierten Ausprägungen enthalten ist. Der Grund dafür ist, dass die meisten diskontinuierlichen Parameter so wesentliche Informationen beschreiben, dass ihre Umsetzvektoren entsprechend große Abstände voneinander haben.

Diese implementierte Lösung ist sicherlich noch verbesserungsfähig. Aus theoretischem Blickwinkel gesehen ist sie zu hart bezüglich weniger wichtigen diskontinuierlichen Parametern, was zu falschen Ergebnissen führen kann, und zu weich bezüglich der kontinuierlichen Parameter, was zu hohen Laufzeiten bei der Prognose führt. In der Praxis spielt der erste Teil keine Rolle, der zweite aber sehr wohl. Insbesondere bei der Aufteilung der Experten nach verschiedenen Werkstoffen, die fast ausschließlich durch kontinuierliche Parameter (Anteile der verschiedenen Legierungselemente) beschrieben werden, sind stets alle Experten zuständig. Aus werkstofftechnischer Sicht unterscheiden sich die Werkstoffe aber je nach Hauptlegierungselement grundlegend.

Natürlich ist die Suche nach robusten Schätzungen oder unteren Schranken für die Prognosefehler die erste Wahl. Da sie jedoch vermutlich schwierig sein wird, soll hier eine alternative Möglichkeit, die Laufzeit bei der Prognose zu verbessern, diskutiert werden[19]. Sie verbessert allerdings nur die Prognosezeit selbst, nicht jedoch die Vorauswahl der Experten und damit die Zeit des Ladens der Experten aus der Datenbank.

Die laufzeit-bestimmende Operation bei der Prognose ist die Berechnung des Prognosefehlers nach Gleichung 3.54, die der Übersichtlichkeit halber hier noch einmal in elementarer Form wiedergegeben ist:

$$\sigma^2(x) \quad = \quad \sum_{m=1}^{M} \frac{1}{\sigma_{\mathrm{w}}^{-2} + \lambda_m} \left(\sum_{i=1}^{M} u_{im} g_i(x) \right)^2 . \tag{5.45}$$

Wie man sieht, besteht die Berechnung aus zwei ineinander geschachtelten Schleifen. Die Summanden der äußeren Summe sind jedoch alle echt positiv, daher können untere Schranken an den Prognosefehler leicht als Teilsummen berechnet werden.

Man kann nun einen Schwellwert θ für den Prognosefehler $\sigma(x)$ definieren und das Netz in der Kooperation vernachlässigen, wenn $\sigma(x) > \theta$ ist. Sortiert man nun die λ_m aufsteigend und beginnt die Summenbildung mit dem kleinsten λ_m, bestehen gute Aussichten, den Schwellwert bereits mit nur wenigen Summanden zu überschreiten, falls $\sigma(x) > \theta$ ist.

Die Verwendung eines derartigen Schwellwerts stellt in der Praxis kein Problem dar. Meist wünscht sich der Benutzer ohnehin, dass ihm allzu große Fehlerbalken überhaupt nicht mehr angezeigt werden.

5.4.9 Ausgangsparameter

Grundsätzlich können alle Ausgangsparameter in einem gemeinsamen Netz, etwa einem Feed-forward-Netz mit mehreren Ausgangsneuronen, trainiert werden. Dabei ist auch eine gemeinsame Regularisierung der Gewichte, die mit den Ausgängen verknüpft sind, sinnvoll, wenn die Ausgangsparameter entsprechend anhand ihrer Sensitivität normiert wurden. Liegt für einen einzelnen Ausgang eines Datensatzes kein Wert vor, so kann er durch einen beliebigen Messwert und den Messfehler ∞ in den Trainingsdaten repräsentiert werden.

In der vorliegenden Implementierung wird allerdings für jeden Ausgangsparameter des konzeptionellen Schemas ein eigenes neuronales Netz nach den Abschnitten 3.1 und 3.2 erstellt. Für dieses Verfahren sprechen zwei Gründe. In den meisten Korrosionssystemen sind nicht alle Ausgangsparameter mit Trainingswert und -fehler belegt, daher unterscheiden sich die Mengen der Trainingsdaten für die einzelnen Ausgangsparameter, was sich wiederum auf die (heuristisch eingestellte) Anzahl der Basisfunktionen auswirkt. Außerdem sind die Trainingsfehler für verschiedene Ausgangsparameter unterschiedlich, sodass unterschiedliche Matrizen A_D entstehen.

Die Netze, die kontinuierlichen Ausgangsparametern zugeordnet sind, besitzen nur einen Ausgang mit Wert und Fehler. Dem gegenüber besitzen die Netze, die diskontinuierlichen Ausgangsparametern zuge-

[19] Sie wurde bisher noch nicht implementiert.

ordnet sind, mehrere Ausgänge für Werte und einen gemeinsamen Ausgangsfehler, was unter Verwendung von Bedingung IIIc (Gleichung 4.121) in Abschnitt 4.3.2 folgt. Bei den diskontinuierlichen Ausgangsparametern ist das Verfahren der speziellen Transformation in Abschnitt 4.3.2 detailliert erläutert, daher beschränkt sich der weitere Abschnitt hier auf kontinuierliche Ausgangsparameter.

Während die Werte der Eingangsparameter softwaretechnisch nur transformiert werden müssen, müssen bei den Ausgangsparametern zusätzlich der Fehler mit- und der Prognosewert und -fehler zurücktransformiert werden. Die Fehler werden beim Training und bei der einfachen Prognose linear approximiert. Ist $\Phi : \mathbb{R} \to \mathbb{R}$ die Transformationsfunktion (hier die Transformation, die Normierung und die Skalierung beinhaltend), t der konzeptionelle Wert, s der konzeptionelle Fehler und t' und s' die entsprechenden transformierten Zahlen, dann wird beim Training

$$t' = \Phi(t) \tag{5.46}$$

$$s' = s \cdot \frac{\partial \Phi}{\partial t}(t) \tag{5.47}$$

und bei der Prognose

$$t = \Phi^{-1}(t') \tag{5.48}$$

$$s = s' \cdot \frac{\partial(\Phi^{-1})}{\partial t'}(t') \tag{5.49}$$

berechnet. Daher müssen neben der Funktion Φ auch ihre Umkehrung Φ^{-1}, ihre erste Ableitung $\partial\Phi/\partial t$ und die erste Ableitung ihrer Umkehrung $\partial(\Phi^{-1})/\partial t'$ implementiert werden.

Da all diese Funktionen existieren müssen, schränkt dies die Wahl möglicher Transformationsfunktionen ein. Sei $R \subseteq \mathbb{R}$ die Menge gültiger Werte des Parameters. Setzt man voraus, dass das Netz keinerlei Forderungen an die Ausgangswerte stellt (etwa bei linearen Ausgangsneuronen), dann muss $\Phi : R \to \mathbb{R}$ nicht nur monoton und stetig differenzierbar wie bei den Eingangsparametern, sondern darüber hinaus auch noch bijektiv sein.

Eine nicht-bijektive Transformationsfunktion ist etwa die sigmoide Transformation $\Phi(t) = 1/(1 + \exp(-t))$, deren Bildmenge nur das Intervall $]0, 1[$ ist. Ein generalisiertes lineares Netz kann jedoch durchaus im Sinne einer Extrapolation einen Prognosewert von $1,1$ berechnen, der aber nicht zurück in das konzeptionelle Schema transformiert werden kann.

Während beim Training die Messfehler in der Regel klein sind und daher gut linear approximiert werden können, können die Prognosefehler groß werden. Daher ist es der Benutzerfreundlichkeit zuträglich, wenn etwa in Grafiken die Prognosefehlerkurven exakt zurücktransformiert werden. Will man etwa das einfache Fehlerintervall darstellen, werden die Kurven $\Phi^{-1}(t' + s')$ und $\Phi^{-1}(t' - s')$ gezeichnet.

Außerdem empfiehlt es sich einen Grenzwert für den Fehler einzuführen und gar keine Kurve mehr zu zeichnen, wenn dieser überschritten wird.

5.4.10 Besondere Transformationen

Nicht immer sind die oben genannten Methoden geeignet, eine ähnlichkeitstreue Transformation der Daten des konzeptionellen Schemas zu leisten.

Ein Beispiel dazu ist etwa die Transformation von Winkelangaben, Tages- oder Jahreszeiten. Diese haben zyklischen Charakter, der natürlich erhalten bleiben sollte. Eine sinnvolle Transformation des Winkels $z \in [0, 2\pi[$ bildet das Paar $(\sin z, \cos z)$. Man beachte, dass sich jede andere Winkelfunktion der Art $\alpha \sin(z + z_0)$ für beliebige $\alpha, z_0 \in \mathbb{R}$ durch Linearkombination, etwa in der ersten Verarbeitungsstufe eines Feed-Forward-Netzes, darstellen lässt.

Kapitel 6

Die Softwareimplementierung

Die bisher vorgestellten Verfahren stellen softwaretechnisch Automatismen ohne Benutzerinteraktivität dar. Demgegenüber beschreibt dieses Kapitel diejenigen Konzepte, die Eingaben von Korrosionsfachleuten erfordern.

Abbildung 6.1 zeigt die Grobstruktur der Implementierung aus Sicht der Benutzer. Das Modul 1 (CORIS) existierte bereits zu Projektbeginn, da es zur Recherche von Korrosionsfakten verwendet wird, und wurde durch [Steinmeier] und [Azizi] erweitert. Da die grundlegenden Methoden während der Projektlaufzeit neu entwickelt wurden, wurde Modul 2 vollständig neu konzipiert und implementiert. Von Modul 3 wurde lediglich ein großer Teil der Oberfläche übernommen, die darunterliegenden Datenstrukturen wurden ebenfalls neu konzipiert und implementiert.

In dieser Arbeit ist vor allem Modul 2 interessant, daher beschränkt sich der Rest dieses Kapitels auf die Beschreibung der Leistungen von Modul 2. Wie die Prognosen in Modul 3 genau berechnet werden, ist bereits durch die vorherigen Kapitel klar beschrieben. Da Modul 3 für Fachleute der Korrosion, die aber kein Fachwissen über neuronale Netze besitzen, entwickelt wurde, ist seine Oberfläche entsprechend aufwendig gestaltet und orientiert sich dabei auch an Besonderheiten der KISS-Datenbank. Es erlaubt die Untersuchung von ganzen Bereichen der Korrosion und enthält umfangreiche Mechanismen zur Darstellung und Optimierung von Prognosen. Details dazu finden sich in [Wendler2].

6.1 Einteilung der Expertenbereiche

Die Festlegung der Bereiche der Experten ist die Kernaufgabe von Modul 2. Da die Korrosionsdaten der KISS-Datenbank prinzipiell Änderungen (im Wesentlichen Neueingaben, aber auch Korrekturen und Löschungen) unterliegen, und das Verfahren der Kooperation nach Abschnitt 4.1 die Disjunktheit der Trainingsdatenmengen verlangt, sind die Trainingsdaten eines Experten allein durch dessen Expertenbereich festgelegt.

Das Erstellen neuer und die Änderung und Löschung vorhandener Expertenbereiche wird durch das in Abschnitt 5.2.6 beschriebene Verfahren ermöglicht. Der Benutzer gibt zunächst einen Anfragebereich A an. Dies kann entweder durch eine explizite Eingabe, einen gespeicherten Anfragebereich, die Vereinigung einiger vorhandener Expertenbereiche oder den Spann von Korrosionssystemen, die in Modul 1 selektiert wurden, erfolgen.

Das System zeigt anschließend alle Korrosionssysteme, die in A liegen, und alle Expertenbereiche, die mit A mindestens überlappen, an. Zu den Korrosionssystemen wird angezeigt, ob und welchem Expertenbereich sie zugeordnet sind. Der Benutzer hat nun zahlreiche Möglichkeiten, Expertenbereiche zu verändern. Dabei werden jedoch nur Änderungen zugelassen, die zu disjunkten Expertenbereichen führen. Ziel ist natürlich, möglichst alle Korrosionssysteme einem Expertenbereich zuzuordnen und dabei die Expertenbereiche so zu wählen, dass die Trainingsdaten eines jeden Experten inhaltlich zueinander passen.

Die Software bietet folgende Operationen auf den Expertenbereichen an:

- *Bearbeiten.* Ein vorhandener Expertenbereich kann manuell in seinen Parameterbereichen verändert

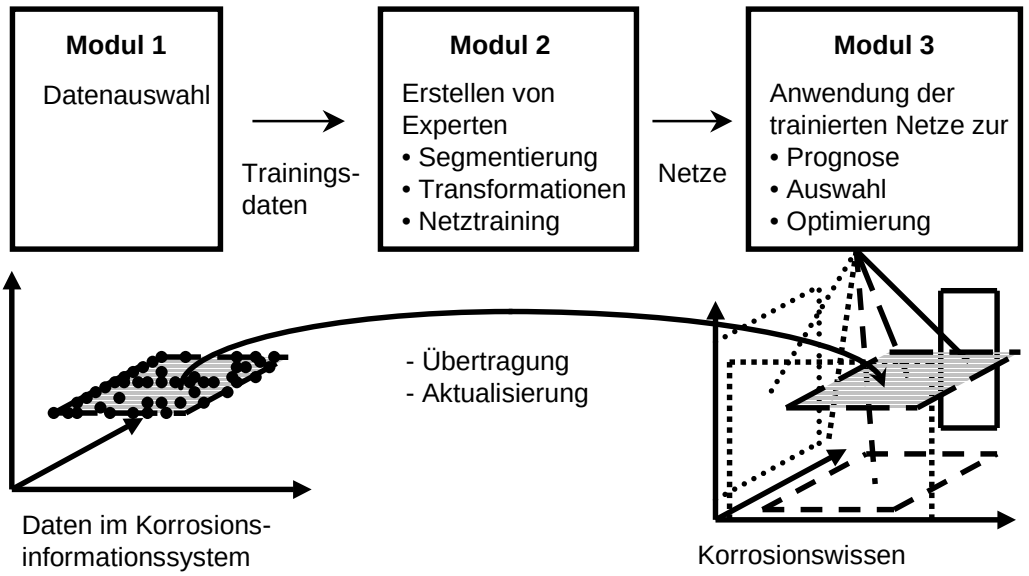

Abbildung 6.1: Übersicht über das Gesamtsoftwaresystem. Die Module 1 und 2 sind in einer gemeinsamen Oberfläche integriert und werden von Korrosionsfachleuten verwendet, die besonders im Umgang mit neuronalen Netzen geschult sind. Modul 3 stellt eine eigenständige Applikation dar und kann von allen Korrosionsfachleuten verwendet werden. Das Diagramm entstammt ursprünglich einer Präsentation von Herrn Schlagner, Bayer AG.

werden. Dabei können nur Parameterbereiche eingegeben werden, die im Anfragebereich liegen. Nach der Bearbeitung wird die Disjunktheit zu anderen Expertenbereichen geprüft.

- *Löschen* eines vorhandenen Expertenbereichs.

- *Teilen.* Ist ein Expertenbereich inhaltlich zu groß oder enthält er für ein effizientes Training zu viele Korrosionssysteme, kann er automatisch anhand eines Parameters in zwei Expertenbereiche geteilt werden. Wird in einem kontinuierlichen Parameter geteilt, wird das Werteintervall in zwei Teilintervalle geteilt. Da die Intervalle immer abgeschlossen sind, entsteht zwischen diesen ein offenes Intervall, das zu keinem Expertenbereich gehört, das aber auch keine Korrosionssysteme enthält. Nach einem diskontinuierlichen Parameter wird geteilt, indem eine echte Teilmenge der Ausprägungen bestimmt wird.
  Das Teilen erfolgt immer inhaltlich anhand der vorhandenen Daten. Im Dialog wird stets angezeigt, welche Parameter sich überhaupt zur Teilung eignen und wie sich die Korrosionssysteme quantitativ auf die neuen Expertenbereiche verteilen werden. Das Teilen ist immer möglich, wenn mehr als ein Datenpunkt im Expertenbereich enthalten ist.

- *Vereinigen* bildet den kleinsten Expertenbereich, der die selektierten Expertenbereiche enthält und ersetzt diese durch die Vereinigung.
  Die Vereinigung von Experten ist nicht immer möglich. Vereinigt man beispielsweise einen Salzsäure- und einen Schwefelsäure-Expertenbereich, erhält man einen Bereich, der beide Bestandteile nur noch optional enthält. Dieser enthält daher auch salz- und schwefelsäurefreie Medien wie etwa reine Salpetersäure, die möglicherweise bereits einem dritten Expertenbereich zugeordnet ist.

- *Neuer Experte für Daten.* Ein neuer Experte kann als Spann von selektierten Korrosionssystemen, die noch in keinem anderen Expertenbereich enthalten sind, konstruiert werden. Der Spann kann ähnlich wie die Vereinigung von Experten nicht immer gebildet werden.

- *Daten zuordnen.* Ein Expertenbereich kann so erweitert werden, dass die selektierten Korrosionssysteme in ihm enthalten sind. Dabei werden zunächst alle Expertenbereiche daraufhin getestet, ob

sie derart erweitert werden können. Dem Benutzer wird dann eine Liste dieser Expertenbereiche präsentiert, die die zu erweiternden Parameter mit ihren neuen und alten Bereichen anzeigt, sodass die Erweiterung inhaltlich so gering wie möglich gehalten werden kann.

Die initiale Festlegung der Expertenbereiche fällt am leichtesten in einer Top-down Strategie. Es wird zunächst ein Expertenbereich erzeugt, der alle Korrosionssysteme der Datenbank enthält. Dieser wird dann sukzessive geteilt. Die ersten Teilungen können aus Effizienzgründen automatisch durch ein heuristisches Verfahren durchgeführt werden.

Die weitere Pflege der Expertenbereiche, insbesondere die Einbringung neu eingegebener Korrosionssysteme, kann inkrementell erfolgen. Dazu werden die neuen Daten, die nicht in bereits existierenden Expertenbereichen enthalten sind, entweder zu neuen Expertenbereichen zusammengefasst oder zu vorhandenen Expertenbereichen durch deren Erweiterung zugeordnet. Man beachte, dass die oben genannten Operationen beide Strategien unterstützen.

Da das konzeptionelle Schema sehr umfangreich ist und sehr viele Korrosionssysteme verarbeitet werden müssen — Anfragebereiche enthalten typischerweise mehrere Tausend Korrosionssysteme und größenordnungsmäßig zehn Expertenbereiche —, wurden Grafiken implementiert, die die Struktur der Korrosionssysteme und der Expertenbereiche untereinander veranschaulichen. Der Benutzer kann dazu eine Menge von Parametern angeben, die er für relevante Kriterien einer Aufteilung hält. Die Software bestimmt nun zu je zwei Korrosionssystemen bzw. Expertenbereichen die Anzahl der angegebenen Parameter, die unterschiedliche Werte bzw. Bereiche besitzen. Diese Anzahl wird dann als inhaltlicher Abstand zwischen den Korrosionssystemen bzw. Expertenbereichen interpretiert. Es wird nun ein Graph berechnet, dessen Knoten die Korrosionssysteme bzw. Expertenbereiche und dessen Kanten die berechneten Abstände bilden. Der Graph wird so dargestellt, dass die euklidischen Abstände der Knoten die Abstände approximieren. Korrosionssysteme bzw. Expertenbereiche, die sich in nur wenigen Parametern unterscheiden, erscheinen so nahe beieinander. Unter Umständen bilden sich Cluster, die vom Benutzer leicht visuell identifiziert werden können. Diese können dann beispielsweise zusammengefasst werden, um sie dem gleichen Expertenbereich zuzuordnen.

6.2 Training der Experten

Nachdem die Bereiche der Experten festgelegt wurden, können die Experten in Modul 2 trainiert werden. Das Training eines Experten ist vollautomatisch und durchläuft folgende Schritte:

1. Laden des Expertenbereichs aus der Datenbank.
2. Laden der Trainingsdaten aus der Datenbank:
   (a) Bestimmen der Rückabbildung der Interpretation und Ausführen in der KISS-Datenbank (Abschnitt 5.3.6).
   (b) Lesen und Interpretieren der Korrosionssysteme aus der Rückabbildung.
   (c) Entfernen der Korrosionssysteme, die nicht im Expertenbereich enthalten sind.
3. Transformieren der Trainingsdaten.
4. Trainieren eines Netzes für jeden Ausgangsparameter.
5. Speichern der Netze in der Datenbank.

Nach erfolgreichem Training werden neben den Netzen noch weitere Informationen in der Datenbank gespeichert. Darunter ist auch der Zeitpunkt des Trainings, der eine regelmäßige Wiederholung des Trainings ermöglicht, um neue und geänderte Daten zu berücksichtigen. Da das Training vollautomatisch ist, kann das Training aller Experten etwa über Nacht erfolgen.

Vor dem Training kann der Korrosionsfachmann festlegen, ob für die kontinuierlichen Ausgangsparameter regionales Rauschen erwartet wird, siehe dazu Abschnitt 4.4.4. Diese Entscheidung könnte er etwa treffen, weil bei einem vorherigen Training einige Trainingsdaten schlecht gelernt wurden.

6.3 Qualität der Daten

Die in der KISS-Datenbank enthaltenen Korrosionssysteme stammen aus verschiedenen Quellen (Labor- oder Betriebsversuche, Literatur) und sind von verschiedenen Mitarbeitern mit unterschiedlichen Intentionen eingegeben worden. Außerdem wurde das KISS-Datenschema im Laufe der Zeit erweitert. Diese Umstände führen dazu, dass die Daten sehr uneinheitlich beschrieben sind, dass bei vielen Korrosionssystemen wichtige Angaben fehlen, und dass viele Korrosionssysteme im Sinne der Interpretation nach Kapitel 5 fehlerhaft sind.

Da die KISS-Datenbank recht umfangreich ist und eine Überprüfung aller Datensätze sehr aufwendig und teuer wäre, wurden jedem Korrosionssystem verschiedene Felder angefügt, die Aussagen zur Qualität, Korrektheit bzw. Vertrauenswürdigkeit enthalten. In ihrer Gesamtheit sind diese Felder in [Azizi] beschrieben.

Das wichtigste Feld ist dabei der Qualitätsstatus, der das Verhältnis des Korrosionssystems zu den Experten beschreibt. Es handelt sich um ein Feld mit folgenden diskreten Werten:

- *Ungeprüft.* Dies ist der initiale Wert, das Korrosionssystem wurde noch nicht trainiert oder nach dem Training nicht manuell geprüft.
- *Unverdächtig.* Das Korrosionssystem wurde mindestens einmal trainiert und geprüft und verhielt sich bei jeder Prüfung unauffällig.
- *Verdächtig.* Das Korrosionssystem wurde geprüft und vom Korrosionsfachmann als verdächtig eingestuft, nachdem Modul 2 eine starke Abweichung zwischen den Trainings- und Prognosewert(en) festgestellt hat.
- *Vermutlich falsch.* Das Korrosionssystem war verdächtig und konnte noch nicht endgültig manuell geprüft werden. Es soll aber derzeit nicht zum Training verwendet werden.
- *Auffällig und korrekt.* Das Korrosionssystem war verdächtig, wurde dann manuell geprüft und für korrekt und vollständig parametrisiert befunden. So können Korrosionssysteme ausgezeichnet werden, die eine Besonderheit, wie etwa einen „Knick“ in der Korrosionsfunktion darstellen. Zwar weicht der prognostizierte Wert stark vom Trainingswert ab, das Korrosionssystem stellt jedoch eine wichtige Information beim Training dar.
- *Ungeeignet.* Das Korrosionssystem war verdächtig, wurde manuell geprüft und ggf. korrigiert, ist aber nicht für das Training mit neuronalen Netzen geeignet und wird daher auch nicht verwendet. Mögliche Ursachen dafür sind Eintragungen in Bemerkungsfeldern, die darauf hinweisen, dass besondere Umstände bei der Messung geherrscht haben, die nicht durch andere Felder einheitlich beschrieben werden können, und die daher ein Korrosionssystem beschreiben, das im konzeptionellen Schema nicht darstellbar ist.
- *Falsch.* Das Korrosionssystem enthält falsche Angaben, soll aber nicht aus der Datenbank entfernt werden.

Der Qualitätsstatus wird — wie im folgenden Abschnitt beschrieben — zum Ausschluss einzelner Korrosionssysteme vom Training verwendet. Er kann aber auch bei einer einfachen Recherche in der KISS-Datenbank angezeigt werden und dient dann als vertrauensbildende Maßnahme für die betroffenen Korrosionssysteme.

In [GuyMatVap] werden automatische Methoden diskutiert, um fehlerhafte Daten zu erkennen. Diese setzen jedoch voraus, dass entsprechend viele, inhaltlich redundante Datensätze vorhanden sind, was für die KISS-Datenbank nicht zutrifft. Daher ist eine manuelle Festlegung — unterstützt durch verschiedene Auswertungen der Software — des Qualitätsstatus unumgänglich.

6.4 Gruppen und Negativlisten

Eine Gruppe ist eine Menge von Experten, jeder Experte gehört zu einer Gruppe. Die Expertenbereiche einer Gruppe unterliegen der Disjunktheitsbedingung und die Experten können nur innerhalb der Gruppe miteinander kooperieren. Die Experten unterschiedlicher Gruppen sind völlig unabhängig voneinander.

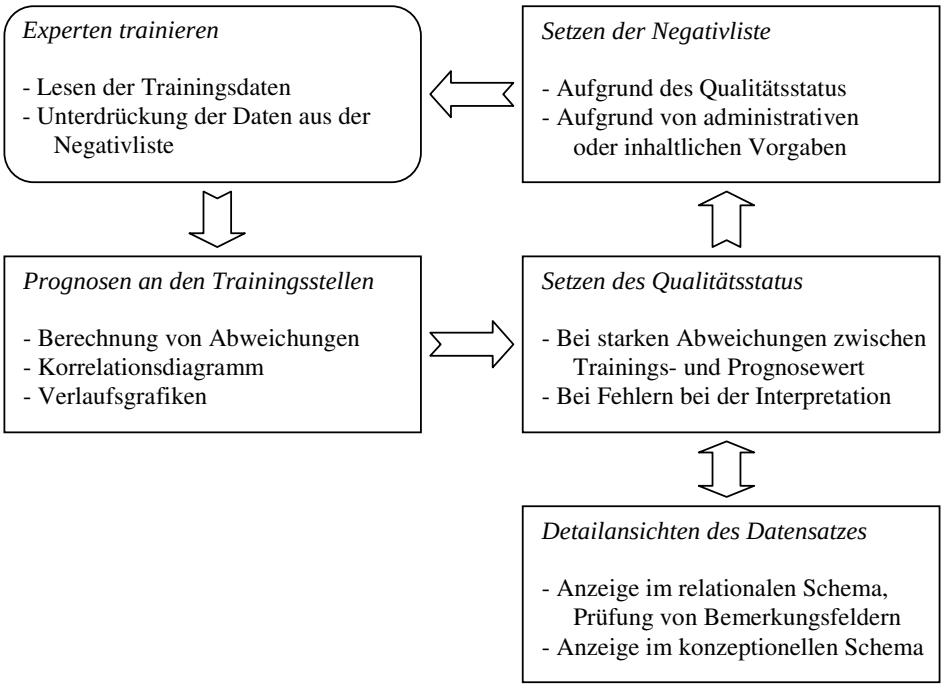

Abbildung 6.2: Schematische Darstellung der logischen Abhängigkeiten zwischen dem Qualitätsstatus, der Negativliste und dem Training. Das Training der Experten (runder Kasten) wird vollautomatisch durchgeführt, alle übrigen Aktionen (eckige Kästen) benötigen manuelle Entscheidungen eines Korrosionsfachmanns.

Das Konzept der Gruppen wurde eingeführt, damit verschiedene Einteilungen der Korrosionssysteme auf Experten gleichzeitig gespeichert und verglichen werden können. Dies ist wichtig, um die Prognosequalität einer Kooperation beurteilen zu können. Außerdem können auch Gruppen erzeugt werden, die nur bestimmte Teilbereiche der Korrosion abdecken, wie es etwa bei abgegrenzten wissenschaftlichen Arbeiten über bestimmte Medien und Werkstoffe der Fall ist.

Jede Gruppe besitzt eine sogenannte Negativliste, die in Form von Verweisen die Menge der Korrosionssysteme umfasst, die nicht zum Training verwendet werden sollen. Die Negativliste wird manuell erzeugt, dabei kann sich der Benutzer am Qualitätsstatus orientieren, er kann aber auch beliebige Korrosionssysteme hinzu- oder herausnehmen.

Abbildung 6.2 zeigt das Zusammenspiel der einzelnen Konzepte zur Qualitätssicherung der Trainingsdaten. Die Erstellung der Negativliste einer Gruppe ist ein zyklischer Prozess, bei dem der Benutzer zwar wesentliche programmtechnische Unterstützung erhält, den er aber letztlich manuell durchführen muss.

Nachdem Experten vollautomatisch[1] trainiert wurden, können Prognosen an den Trainingsstellen berechnet werden. Eine starke Abweichung zwischen Trainings- und Prognosewert kann dabei ein Indiz für eine Fehleingabe des Korrosionssystems sein. Ein Maß für eine zu starke Abweichung gibt es aber nicht, vielmehr muss im Einzelfall und mit Korrosionsfachwissen entschieden werden.

Mit dieser Methode kann selbstverständlich nur ein kleiner Teil der Fehler in der Datenbank entdeckt werden. Insbesondere werden fehlerhafte Korrosionssysteme nicht entdeckt, die weit abseits anderer Messstellen liegen. Auch umgekehrt ist eine große Abweichung zwischen Trainings- und Prognosewert noch kein Beweis für einen Fehler, sondern vielmehr nur ein Verdacht, wie er in einigen Ausprägungen des Qualitätsstatus bezeichnet wird.

Aufgrund der Abweichungen kann für die untersuchten Trainingsdaten ein entsprechender Qualitätsstatus gesetzt werden. Problematisch ist dabei möglicherweise die Verknüpfung der Ergebnisse verschiedener Gruppen. Prinzipiell kann sich ein Korrosionssystem in einer Gruppe völlig unauffällig verhalten,

[1] mit Ausnahme der Angabe, ob regionales Rauschen möglich ist, s.o.

während in einer anderen Gruppe eine sehr starke Abweichung auftritt. Hier ist ebenfalls eine manuelle Untersuchung durch einen Korrosionsfachmann notwendig und sinnvoll.

Der Qualitätsstatus kann nicht nur anhand der oben genannten Abweichungen, sondern auch aufgrund anderer Auffälligkeiten und Untersuchungen gesetzt werden, jedoch immer nur manuell. Dies betrifft insbesondere automatisch festgestellte Fehler bei der Interpretation der Ursprungsdaten in das konzeptionelle Schema (Abschnitt 5.3) und den Inhalt von Bemerkungsfeldern in der KISS-Datenbank.

Die Negativliste kann nun aufgrund des Qualitätsstatus erstellt bzw. verändert werden. Man beachte, dass der Qualitätsstatus unabhängig von den Gruppen, die Negativliste aber individuell für jede Gruppe ist. Selbstverständlich kann die Negativliste auch direkt verändert werden.

Mit den vorgestellten Konzepten kann ein Qualitätsmanagement der Korrosionsdaten durchgeführt werden. Dabei ist natürlich die Korrektur von Fehlern oder die Anpassung der Interpretation vorrangig gegenüber dem Eintrag in der Negativliste. Diesem Ziel steht aber der Arbeitsaufwand für das manuelle Überprüfen der Korrosionssysteme gegenüber, der ganz erheblich sein kann. Folglich enthält der Qualitätsstatus eben auch die Ausprägungen *verdächtig* und *vermutlich falsch*, die nur temporär, also bis zum Zeitpunkt einer manuellen Prüfung, verwendet werden sollten.

Kapitel 7

Ergebnisse

Das Gesamtsystem, das aus den bisher beschriebenen Komponenten besteht, wurde in verschiedenen Tests empirisch auf seine Korrektheit und Leistungsfähigkeit hin untersucht, die Ergebnisse werden hier vorgestellt. Es sei darauf hingewiesen, dass einige empirische Untersuchungen zu Teilkonzepten bereits zusammen mit ihrer mathematischen Beschreibung aufgeführt worden sind, in diesem Kapitel geht es also nur um das Gesamtsystem.

7.1 Verteilung der KISS-Daten

Um die nachfolgenden Untersuchungen interpretieren zu können, ist es nötig, sich einen kurzen Überblick über die Struktur der Daten in der KISS-Datenbank zu verschaffen.

Das Medium gilt in der Korrosion als wichtigstes Unterscheidungsmerkmal von Korrosionssystemen. Dabei werden die Medien nach ihren Bestandteilen klassifiziert, Abbildung 7.1 zeigt eine Übersicht. Man sieht, dass die Intensität der Untersuchungen einzelner Medien extrem unterschiedlich ist. Zwar sind in der Datenbank insgesamt etwa 2400 verschiedene Medienbestandteile aufgeführt, jedoch lassen sich bereits knapp die Hälfte aller Medien als Gemisch aus je einer von sieben Säuren und Wasser beschreiben. Die anderen Medien unterscheiden sich stark in ihrer Beschreibung: bei einigen liegt keinerlei Information über ihre Bestandteile vor, sie werden daher nur über ihren sogenannten Hauptnamen, ein frei einzugebender Text, beschrieben. Bei anderen ist offensichtlich, dass einige, aber nicht alle Bestandteile eingegeben wurden, sie werden gesondert im konzeptionellen Datenschema dargestellt. Weitere grobe Klassifizierungsmerkmale der Medien ist die Anzahl ihrer Bestandteile und das Enthaltensein von Wasser.

In Abbildung 7.1 werden bestimmte Medienklassen durch ihre Farbe/Schraffur unterschieden: je heller, desto mehr Information liegt vor und desto besser kann aus ihr generalisiert werden. Medien aus Schwefelsäure und Wasser etwa wurden sowohl intensiv untersucht, als auch spannen sie nur einen kleinen Teilraum des Eingangsraums, nämlich nur eine Medien-Dimension, auf. Diese beiden Eigenschaften sind günstig für eine gute Generalisierung. Genau umgekehrt sieht es bei den Medien aus, die als einzige Information ihren Hauptnamen besitzen: sie wurden oftmals nur einmalig oder in einer kleinen Versuchsreihe vermessen, sie besitzen kein sinnvolles Ähnlichkeitsmaß zu anderen Medien, und sie spannen durch ihre 1-aus-k-Kodierung des Hauptnamens einen sehr großen Eingangsraum auf. Trotzdem ist es sinnvoll auch sie zu trainieren, wenn innerhalb einer Messreihe, beispielsweise über eine Variation der Temperatur, generalisiert werden soll.

Das Medium ist zwar für die Korrosion sehr wichtig, jedoch nicht alleine ausschlaggebend. Auch die Werkstoffe unterscheiden sich fundamental in ihrer Korrosionswirkung, weshalb auch eine entsprechende Klassifizierung der Werkstoffe in der Werkstofftechnik vorgenommen wird. Datentechnisch unterscheiden sich Medium und Werkstoff aber sehr, denn als Legierungsbestandteile kommen nur einige chemische Elemente, derzeit etwa 40 verschiedene, in Betracht. Die Anzahl derer, die tatsächlich in einem konkreten Werkstoff vorkommt, ist aber deutlich höher als beim Medium: während beim Medium in der Regel nur zwei oder drei Bestandteile beschrieben sind, sind es beim Werkstoff typischerweise etwa zehn. Daher fällt eine harte Klassifizierung beim Werkstoff wesentlich schwerer als beim Medium.

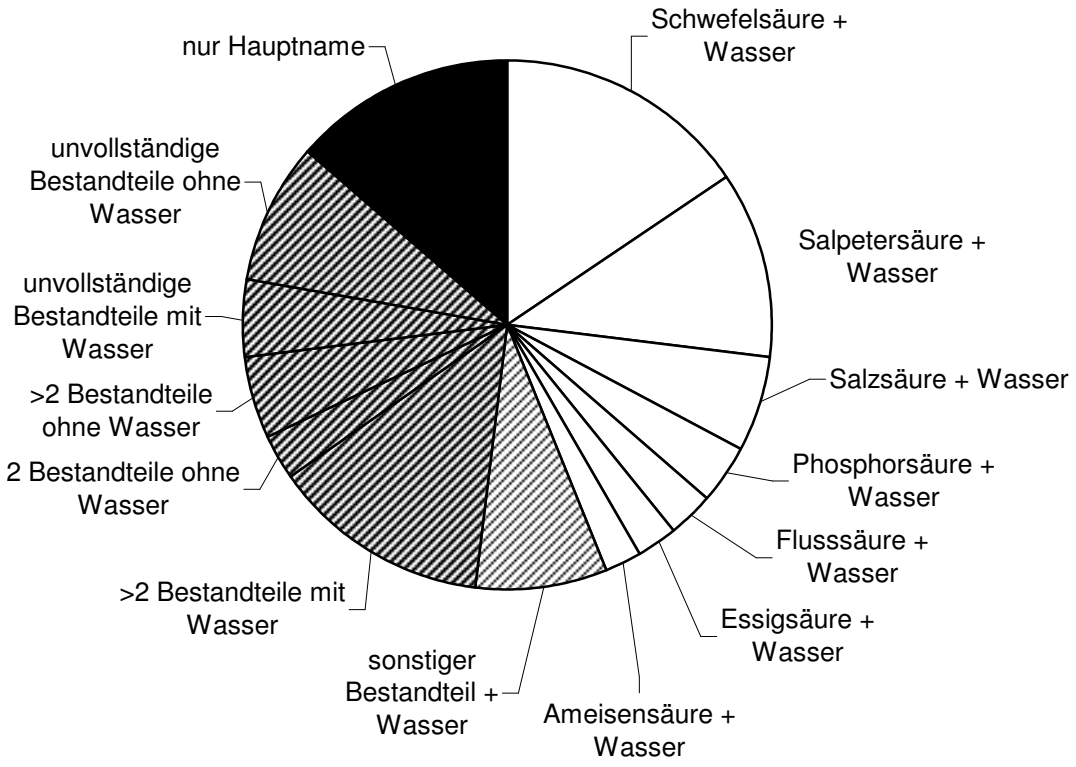

Abbildung 7.1: Verteilung der Medien in der KISS-Datenbank

Interpretiert man die Farbgebung der Abbildung 7.1 bezüglich der Generalisierungsfähigkeit in bestimmten Bereichen der Datenbank, so wird sich ein ähnliches Bild für den Werkstoff und ein weiteres ähnliches Bild für die übrigen korrosionsbestimmenden Parameter ergeben. Da die entsprechenden Gruppen natürlich überlappen, bleiben nur wenige Datensätze übrig, die in Bereichen liegen, die in allen drei Bildern zu den weißen Flächen gehören würden. Mit diesen Überlegungen soll deutlich gemacht werden, wie gering die Erwartungen an die Generalisierungsfähigkeit des Softwaresystems aufgrund der vorhandenen Daten sind.

7.2 Vergleich verschiedener Einteilungen

Die Einteilung der vorhandenen Trainingsdaten aus der KISS-Datenbank auf die einzelnen Experten wird bekanntlich manuell unter Berücksichtigung von Kossosionsfachwissen durchgeführt. Dieses Fachwissen ist allerdings nur schwer auf die Festlegung von disjunkten Bereichen im konzeptionellen Schema zu übertragen. Es ist daher zu hoffen — und die Erkenntnisse aus Abschnitt 4.2 geben Anlass dazu — dass die genaue Experteneinteilung die Prognosen bzw. die Generalisierungsfähigkeit des Gesamtsystems nicht wesentlich mitbestimmt.

Der Einfluss der Einteilung wurde daher wie folgt untersucht. Aus der gesamten KISS-Datenbank wurden alle Messungen aus einem etwa viermonatigen Zeitraum selektiert. Diese gliedern sich in 100 sogenannte Vorgänge, die eine administrative Struktur darstellen und in der Regel eine bestimmte Untersuchungsreihe enthalten. Die 50 geraden Vorgangsnummern bildeten das Lernset mit 2314 Korrosionssystemen, die 50 ungeraden Vorgangsnummern das Testset mit 1986 Korrosionssystemen. Diese über Vorgänge gesteuerte Lern/Testset-Einteilung stellt natürlich eine besondere Herausforderung für das Gesamtsystem dar, da so Lern- und Testset sehr verschiedene Cluster im Eingangsraum beschreiben und eine Generalisierung daher besonders schwierig ist. Wäre die Lern/Testset-Einteilung zufällig auf Ebene einzelner Korrosionssysteme erfolgt, würde dies weniger das Gesamtsystem, sondern vielmehr einzelne

Nr.	Art der Einteilung	Anzahl der Experten	Bemerkung
1	nur Druck	2	Verteilung der Daten: 2241 zu 73
2	nach Werkstoff	12	
3	nach Belastung	8	Temperatur trennt drei Teilbereiche
4	nach Medium	6	ein Experte wurde im Druck expandiert
5	Medium + Druck	9	verfeinerte Einteilung von Nr. 4
6	Medium + Werkstoff	18	verfeinerte Einteilung von Nr. 5
7	M+D, 2*Gewichte	9	wie Nr. 5, alle Experten haben doppelte Anzahl Basisfunktionen

Tabelle 7.1: Übersicht über die getesteten Einteilungen. Die genaue Parametrisierung der Einteilungen findet sich in der Bayer-internen Dokumentation.

Netze auf ihre Generalisierungsfähigkeit prüfen, dies ist jedoch bereits in Abschnitt 3.4.2 erfolgt. Zudem entspricht eine vorgangsweise Prognoseanfrage der typischen Anwendungssituation: Ziel ist (unter anderem) Laborversuche zu vermeiden. Daher sind Anfragen gerade abseits der vorhandenen Messreihen wahrscheinlich.

Tabelle 7.1 stellt nun die wichtigsten Eigenschaften der getesteten Einteilungen der Korrosionssysteme des Lernsets dar. Einteilung 1 dient als Referenz und beschreibt im Wesentlichen einen universellen Experten. Lediglich die 73 Datensätze, die einen konkreten Druck besitzen, wurden von denen mit verteiltem Druck abgetrennt und in einem eigenen Experten trainiert, um eine Explosion der transformierten Trainingsdaten nach Abschnitt 5.4.7 zu vermeiden. Natürlich wird erwartet, dass Einteilung 1 die beste Generalisierungsfähigkeit besitzt.

Zunächst wurden Einteilungen untersucht, die sich an den drei Kategorien der Eingangsparameter orientieren: der Belastung, dem Werkstoff und dem Medium. Dabei wurde eine Top-down-Strategie verwendet, die iterativ trennende Parameter sucht, die eine sinnvolle Abspaltung von zusammenpassenden Korrosionssystemen ermöglicht. Aufgrund der starken Clusterung der Lernsetdaten gibt es natürlich faktische Korrelationen zwischen diesen Einteilungen.

Während die Einteilung beim Medium (Nr. 4) fünf häufige Bestandteilskombinationen abtrennt und die übrigen Korrosionssysteme in einem sechsten Expertenbereich zusammenfasst, mussten beim Werkstoff (Nr. 2) und bei der Belastung (Nr. 3) kontinuierliche Parameter zur Trennung verwendet werden. Beim Werkstoff waren gewisse Cluster in den Trainingsdaten erkennbar (wie auch in der gesamten Datenbank), die eine auf gewisse Weise natürliche Einteilung ermöglichten. Bei der Belastung fehlten entsprechende Cluster nach der Abtrennung von bestimmten Sonderfällen der Belastung jedoch völlig, weshalb der Temperaturbereich recht willkürlich in drei Intervalle und damit drei Experten unterteilt wurde.

Da das Medium die natürlichste Einteilung erlaubt, wurde die Einteilung Nr. 4 noch in zwei Schritten (Nr. 5 und 6) verfeinert, um den Einfluss der Einteilungsgranularität untersuchen zu können. Schließlich wurde noch in Einteilung 7 der Einfluss von deutlich mehr Gewichten (Basisfunktionen) in den einzelnen Experten untersucht. Diese könnten wesentlichen Einfluss auf die Gewichtsregularisierung und damit das Extrapolationsverhalten der Experten besitzen.

In Abbildung 7.2 sind die Zeiten zum Training aller Experten einer Einteilung und zur Prognose an allen Stellen des Testsets dargestellt. Die Trainingszeit umfasst dabei nicht nur das eigentliche Training mit der asymptotischen Komplexität $O(N^3)$ (N = Anzahl der Basisfunktionen $\approx$ Anzahl der Trainingsdaten) jedes Experten, sondern auch die Rückabbildung des Expertenbereichs auf das relationale KISS-Datenschema (Abschnitt 5.3.6) sowie das Laden der Trainingsdaten aus der KISS-Datenbank. Bei Einteilung 1 (nur Druck) spielt dabei klar das Netztraining die Hauptrolle, daher ist in der Praxis die Verwendung nur eines Experten für alle Korrosionssysteme der KISS-Datenbank aus Zeitgründen unmöglich. Betrachtet man die Einteilungen 4 (nach Medium), 5 (Medium + Druck) und 6 (Medium + Werkstoff), die eine Folge von Verfeinerungen darstellen, so dominiert bei Nr. 4 die Netztrainingszeit des einen Experten, dessen Trainingsdaten teils im Druck expandiert werden mussten. Bei Einteilung 6 dominiert dagegen die Zeit, die für die Rückabbildung benötigt wird, da sehr viele Experten in der

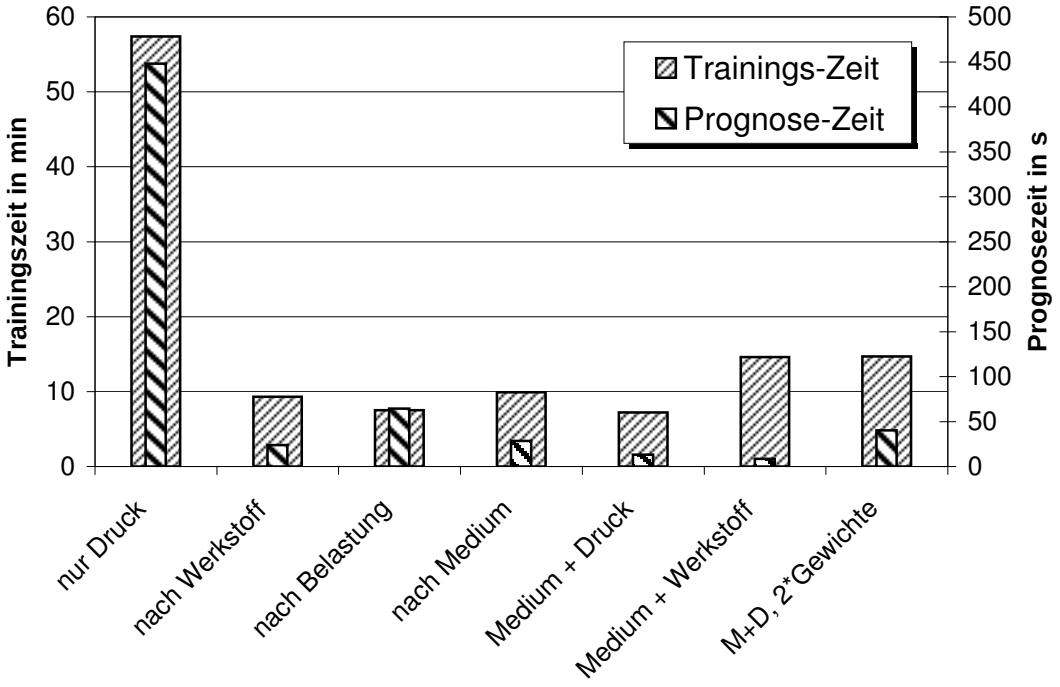

Abbildung 7.2: Zeitvergleiche der verschiedenen Einteilungen

Einteilung vorliegen. Einteilung 5 stellt offensichtlich hier einen günstigen Kompromiss zwischen vielen und wenigen Experten dar. Alles in allem sind die absoluten Trainingszeiten jedoch sehr günstig, eine Einteilung der gesamten Datenbank kann in wenigen Stunden, etwa über Nacht, trainiert werden.

Anders sieht dies bei den Prognosezeiten aus. Hier ist zu bedenken, dass in der praktischen Anwendung sehr viele Prognosen berechnet werden müssen, beispielsweise bei der graphischen Darstellung von Verläufen oder der algorithmischen Optimierung des Korrosionsverhaltens. Die Prognosezeit setzt sich aus dem Zuständigkeitstest (Abschnitt 5.4.8), dessen Laufzeit aufgrund des komplexen konzeptionellen Datenschemas nicht zu vernachlässigen ist, und bei Zuständigkeit der eigentlichen Laufzeit zur Netzprognose zusammen. Bei Einteilung 1 (nur Druck) dominiert hier klar die Netzprognosezeit, ihre Höhe ist ein Argument gegen die Verwendung eines einzelnen Experten für die gesamte Datenbank. Einteilung 3 (Belastung) orientiert sich wesentlich an der Temperatur. Da diese ein kontinuierlicher Parameter ist, sind die entsprechenden Experten immer gemeinsam zuständig, was dazu führt, dass für sie immer gemeinsam Netzprognosen berechnet werden müssen. Die Expertenzuständigkeit ist also in dieser Einteilung wenig selektiv und erhöht daher die Prognosezeiten. Bei allen übrigen Einteilungen sieht man die Tendenz zu geringeren Prognosezeiten bei höherer Anzahl von Experten.

Wichtiger als die Rechenzeiten sind natürlich die Prognosen selbst. Für alle Stellen des Testsets wurden Prognosen berechnet und anhand des Prognosefehlers nach Tabelle 7.2 klassifiziert[1]. Die Abtragungsge-

[1] Die tatsächliche Klassifizierung basiert auf den transformierten Fehlern der Abtragungsgeschwindigkeit. Die verwendeten Klassifikationsgrenzen sind 1 und 3, die in Tabelle 7.2 dargestellten Grenzen $2,7 \approx \exp(1)$ und $20 \approx \exp(3)$ gelten daher genau genommen nur im logarithmischen Teil der Transformationsfunktion der Abtragungsgeschwindigkeit.

brauchbar	relativer Prognosefehler der Abtragungsgeschwindigkeit kleiner als $2,7$
bedingt brauchbar	relativer Prognosefehler der Abtragungsgeschwindigkeit zwischen $2,7$ und 20
unbrauchbar	relativer Prognosefehler der Abtragungsgeschwindigkeit größer als 20
nicht prognostiziert	kein Experte war zuständig

Tabelle 7.2: Klassifizierung der Testset-Prognosen anhand des Prognosefehlers

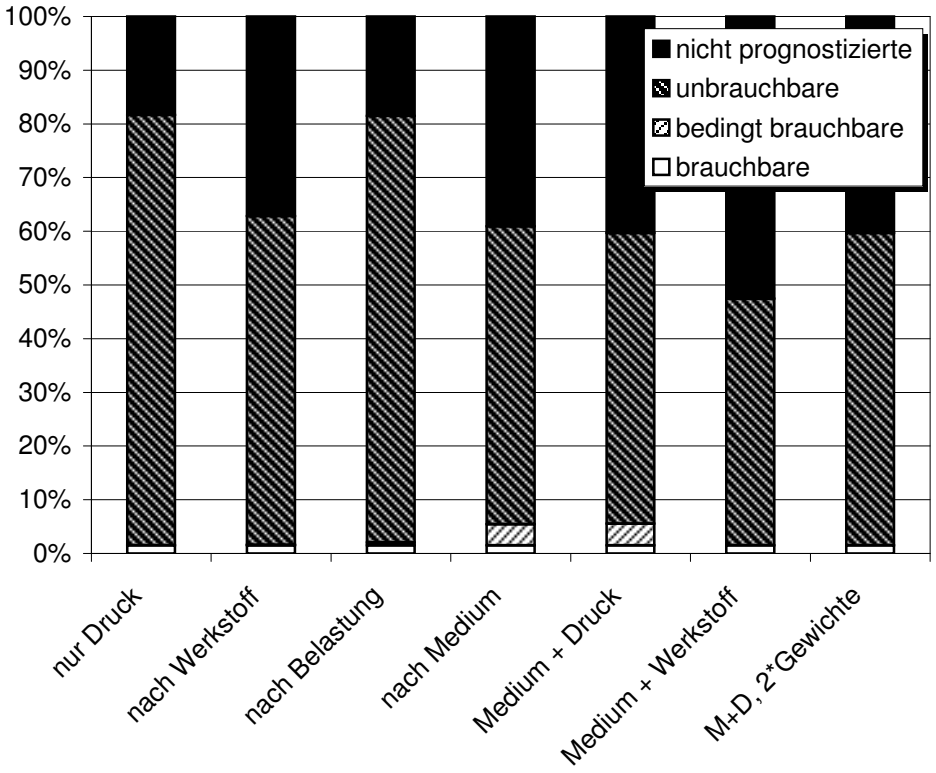

Abbildung 7.3: Verteilung der Brauchbarkeit der Prognosen bei verschiedenen Einteilungen

schwindigkeit gilt dabei als wichtigste Ausgangsgröße der Korrosion, typische Werte erstrecken sich dabei über mehrere Zehnerpotenzen. Für viele praktische Zwecke ist bereits die Bestimmung der Größenordnung ausreichend, daher wurden bei der Klassifizierung groß anmutende Prognosefehlergrenzen (Faktor $2,7$ bzw. Faktor 20) zugrunde gelegt.

Abbildung 7.3 zeigt nun die Verteilung der Prognosefehlerklassen bei den verschiedenen Einteilungen. Die Menge der brauchbaren Prognosen im Testset ist bei allen Einteilungen nahezu gleich, sie entstammen bei allen Einteilungen dem gleichen Vorgang (der Vorgang enthält aber noch weitere Korrosionssysteme). Dass sie vergleichsweise klein ist, liegt am relativ ungünstigen Verhältnis zwischen Eingangsdimension und Anzahl der Datensätze im Lernset.

Die Menge der nicht prognostizierbaren Stellen des Testsets bestimmt sich ausschließlich über die diskontinuierlichen Eingangsparameter und ist daher zwangsweise bei Einteilung 1 minimal. Die anderen Einteilungen schränken die Menge der Kombinationen von Ausprägungen verschiedener Parameter ein, sodass einige Kombinationen des Testsets nicht mehr prognostizierbar sind. Dies ist aber für die Praxis ohne Belang, denn es findet lediglich ein Wechsel zwischen den Klassen nicht prognostizierbar und unbrauchbar statt.

Interessant ist die Menge der bedingt prognostizierbaren Stellen. Sie ist in signifikanter Anzahl nur bei den Einteilungen 4 (nach Medium) und 5 (Medium + Druck) vorhanden, insbesondere nicht im globalen Experten 1 (nur Druck) und auch nicht in Einteilung 7 (doppelte Gewichte), deren Bereiche identisch mit Nr. 5 sind. Aus Abbildung 7.3 alleine geht nicht hervor, ob sich die Einteilungen 4 und 5 hier überschätzen, also einen zu geringen Prognosefehler berechnen, oder ob sich die anderen Einteilungen unterschätzen.

Darüber geben aber die Abbildungen 7.4 und 7.5 Auskunft. Sie vergleichen die Differenz zwischen dem Prognose- und dem Trainingswert und setzen ihn ins Verhältnis zum stochastisch erwarteten Fehler: die Größe

$$\delta_n \quad := \quad \frac{t_n - \mu(x_n)}{\sqrt{s_n^2 + \sigma^2(x_)}} \tag{7.1}$$

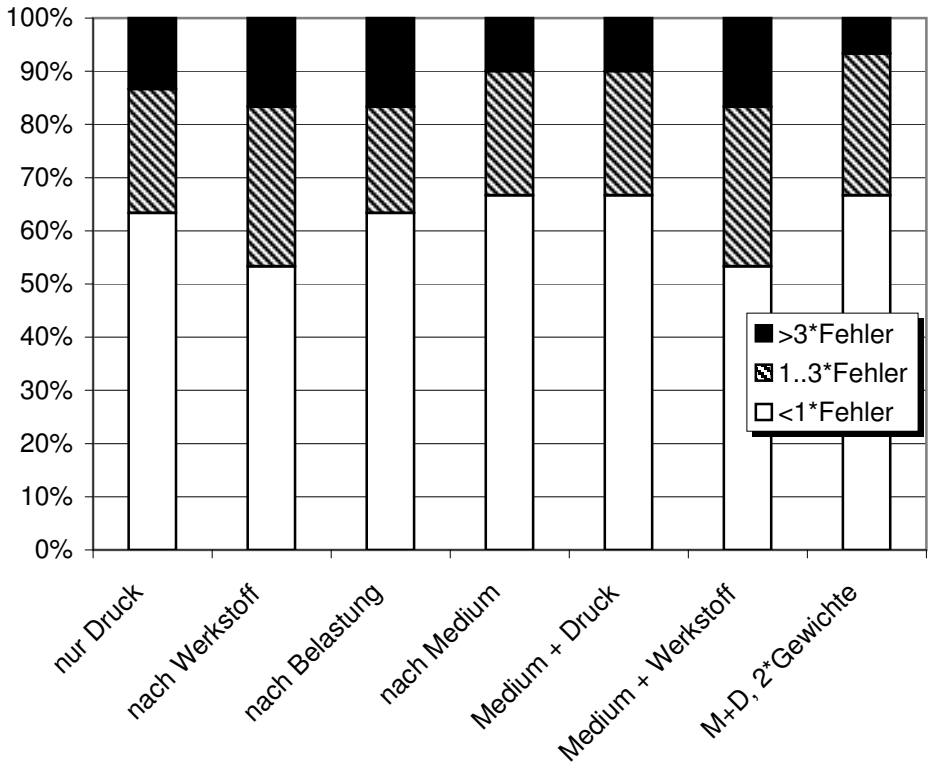

Abbildung 7.4: Generalisierung der brauchbaren Prognosen bei verschiedenen Einteilungen

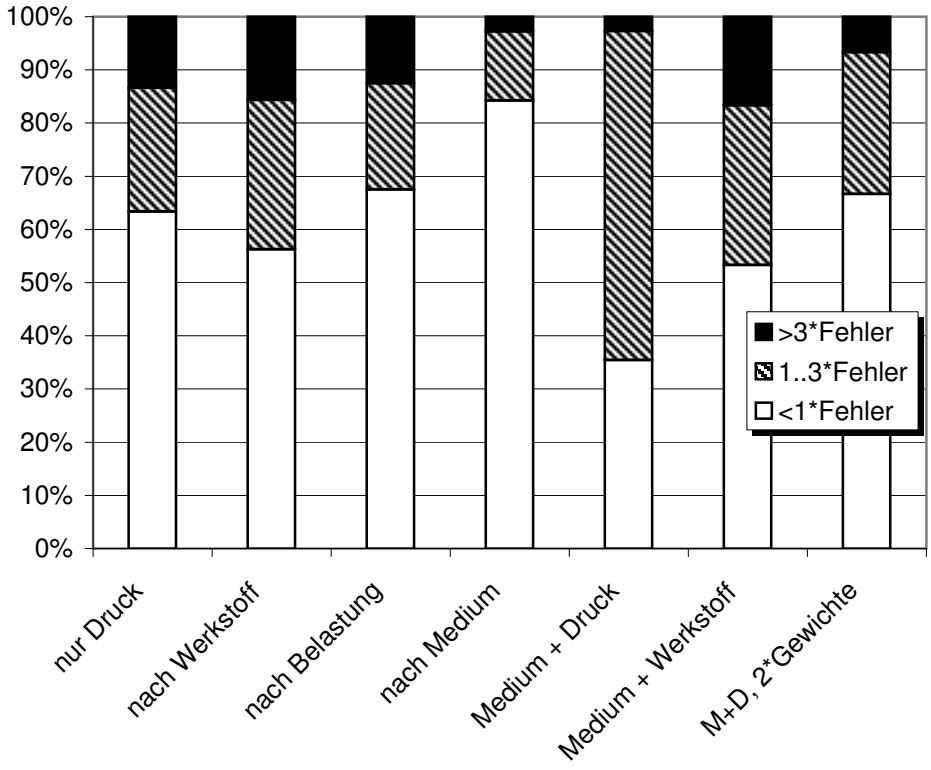

Abbildung 7.5: Generalisierung der brauchbaren und bedingt brauchbaren Prognosen

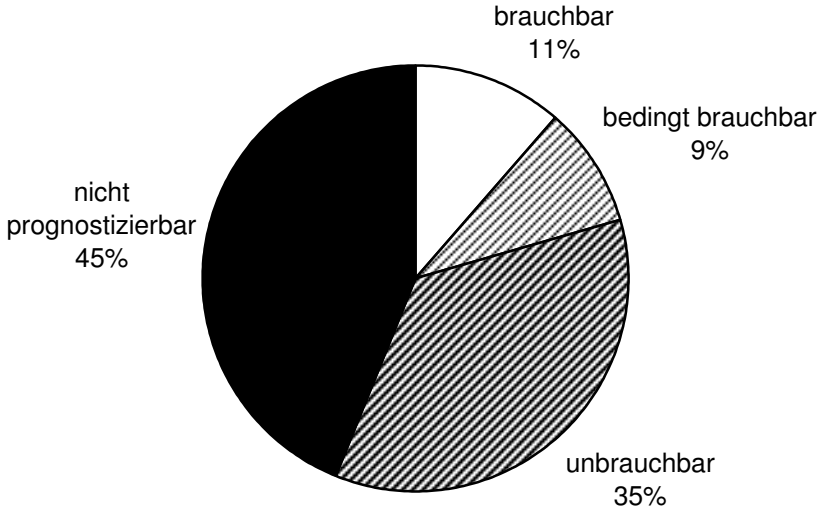

Abbildung 7.6: Brauchbarkeit der Prognosen einer globalen Einteilung

sollte standard-normalverteilt sein, wenn die Prognose korrekt berechnet wurde.

In den Abbildungen 7.4 und 7.5 ist nun die Verteilung der $|\delta_n|$, eingeteilt in je drei Klassen, dargestellt. Bei den brauchbaren Prognosen ist diese Verteilung für alle Einteilungen zufriedenstellend[2]. Die Abhängigkeit der Verteilung unter den verschiedenen Einteilungen ist eher gering. Bei den mindestens bedingt brauchbaren Prognosen stellen die Einteilungen 4 (nach Medium) und 5 (Medium + Druck) Ausreißer nach oben bzw. unten dar. Es konnte keine Begründung gefunden werden, es wird aber vermutet, dass es sich um statistische Ausreißer aufgrund der geringen Anzahl der Datensätze handelt.

Insgesamt ist zu erkennen, dass die Einteilung nur geringen Einfluss auf die Güte der Prognosen hat. Somit bleibt als Hauptargument im Vergleich von verschiedenen Einteilungen vor allem die Laufzeit bei der Prognose.

7.3 Test der globalen Generalisierungsfähigkeit

Zwar hat der Vergleich verschiedener Einteilungen bereits einen kleinen Einblick in die Leistungsfähigkeit des Systems ermöglicht, dieser ist jedoch aufgrund der geringen Menge der gelernten und getesteten Daten nicht repräsentativ. Daher wurden alle Daten der Datenbank vorgangsweise in ein Lernset (ungerade Vorgangsnummern, 40008 Korrosionssysteme) und ein Testset (gerade Vorgangsnummern, 37501 Korrosionssysteme) aufgeteilt. Auf dem Lernset wurden 63 Experten eingeteilt: primär nach den Medienbestandteilen, danach nach Werkstoffeigenschaften. Eine Expansion von Trainingsdaten aufgrund von verteilten und konkreten Werten eines Parameters in einem Experten wurde vermieden.

Abbildung 7.6 zeigt die Brauchbarkeit der Prognosen, eingeteilt in Klassen nach Tabelle 7.2. Die tatsächliche Anzahl brauchbarer und bedingt brauchbarer Prognosen dürfte dabei noch geringfügig höher ausfallen, wenn im praktischen Einsatz die gesamte Datenbank (Lernset + Testset) zum Training zur Verfügung steht. Die Anteile der brauchbaren und bedingt brauchbaren Prognosen sind wesentlich bedingt durch die Dimension des Eingangsraums, durch Clusterbildung in ihm und natürlich die Anzahl eingegebener Korrosionssysteme. Man vergleiche sie daher mit den in Abschnitt 7.1 angestellten Überlegungen. Im übrigen sind sie weniger aussagekräftig für die Beurteilung der vorliegenden Arbeit, sondern beschreiben mehr wirtschaftliche Kenndaten: etwa 11..20% der Korrosionsversuche könnten prinzipiell in Zukunft nur durch den Einsatz des Softwaresystems vermieden werden, und dieser Anteil steigt mit jedem eingegebenen Korrosionssystem.

Die Abbildungen 7.7 bis 7.9 zeigen nun die fehler-relativen Abweichungen zwischen Trainings- und Prognosewert ($|\delta_n|$ nach Gleichung 7.1) auf dem Testset. Diese empirisch gefundenen Verteilungen lassen

[2]Eine detailliertere Diskussion folgt in Abschnitt 7.3

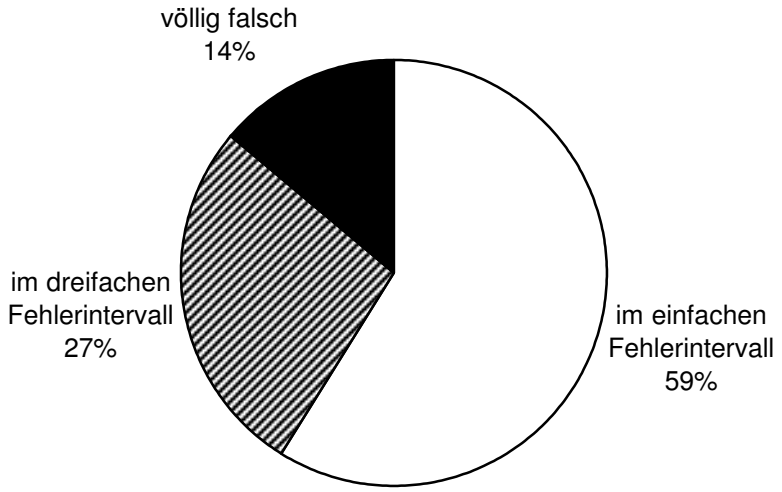

Abbildung 7.7: Generalisierung der brauchbaren Prognosen

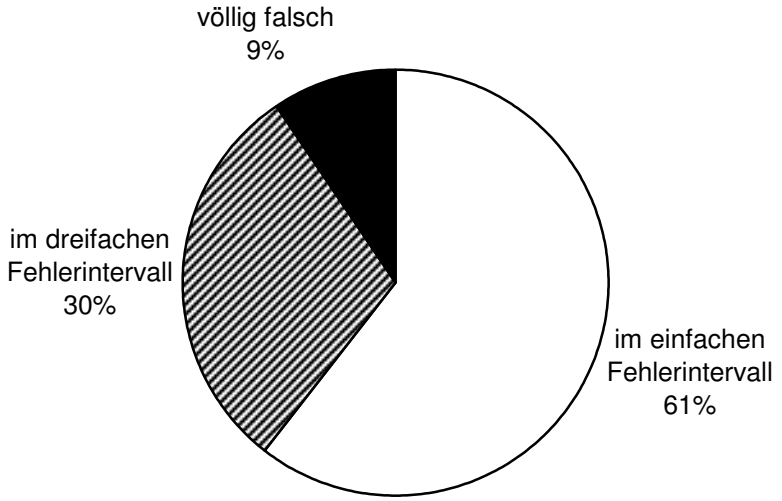

Abbildung 7.8: Generalisierung der (genau) bedingt brauchbaren Prognosen

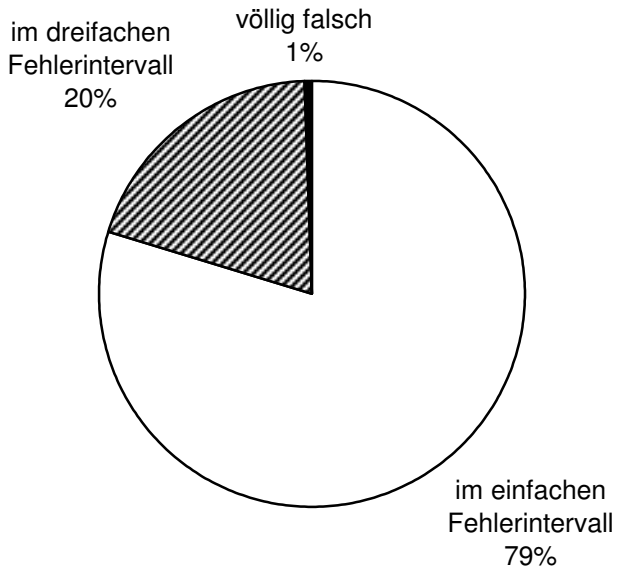

Abbildung 7.9: Generalisierung der unbrauchbaren Prognosen

sich mit den Fraktilen der Standard-Normalverteilung vergleichen. Demnach sollten $\Phi(1) - \Phi(-1) \approx 68,26\%$ der Abweichungen im einfachen Fehlerintervall und $\Phi(3) - \Phi(-3) \approx 99,74\%$ ein dreifachen Fehlerintervall liegen.

Bei allen drei dargestellten Brauchbarkeitsklassen ist dies zufriedenstellend gut erreicht. Zwar gibt es erheblich mehr Ausreißer (schwarze Segmente) als statistisch erwartet, dies lässt sich jedoch leicht durch die noch zahlreich vorhandenen Eingabefehler in den Daten erklären. Man beachte dabei, dass diese Fehler sowohl das Lern- als auch das Testset betreffen. Sind insgesamt etwa 10% aller Daten fehlerhaft; folgen daraus — grob gerechnet — 10% objektiv fehlerhafte Prognosen, zusammen mit den 10% fehlerhaften Messwerten im Testset folgen daraus etwa 20% stark abweichende Datensätze. Der tatsächliche Anteil fehlerhafter Datensätze ist natürlich kaum ermittelbar, erfahrungsgemäß liegt er aber über 10%.

Bei den unbrauchbaren Prognosen liegen deutlich mehr Datensätze im einfachen Fehlerintervall als statistisch erwartet, die Prognosefehler sind also im Vergleich zu den Abweichungen zwischen Mess- und Prognosewert zu groß. Man würde sich daher kleinere Prognosefehler wünschen. Dabei gibt es nun zwei Konzepte, die in diesem Zusammenhang zu untersuchen sind:

- Die Basisfunktionen wurden so gewählt, dass sie außerhalb des Netzbereichs asymptotisch linear divergieren. Diese Wahl wurde dadurch begründet, dass die Prognosefehler außerhalb des Netzbereichs groß bzw. ansteigend sein sollten. Man könnte nun aufgrund der Abbildung 7.9 vermuten, dass dieser Anstieg zu groß sein könnte. Dazu müsste empirisch überprüft werden, ob Basisfunktionen, die etwa gegen eine nicht verschwindende Konstante konvergieren, zu besseren Ergebnissen führen.

- Die Behandlung von Trainingsdaten, die in einzelnen Parametern singuläre (Abschnitt 5.4.6) oder verteilte (Abschnitt 5.4.7) Werte aufweisen, besteht aus heuristischen Methoden, die bei der Prognose den Prognosefehler des Netzes um weitere Komponenten ergänzen. Diese Methoden basieren auf der Parametersensitivität. Zu überprüfen ist hier, ob die Parametersensitivitäten gut gewählt sind, und ob die darauf aufbauenden Konzepte nicht verbessert werden können.

Die geübte Kritik sollte allerdings nicht überbewertet werden. Sie betrifft ohnehin nur Prognosen mit riesigem Prognosefehler, die zwar bei verbessertem Gesamtsystem kleiner, aber wahrscheinlich nicht brauchbarer werden. Von daher ist der praktische Nutzen derartiger Untersuchungen und Verbesserungen fraglich.

Kapitel 8

Schlussbetrachtungen

8.1 Zusammenfassung

Zu zahlreichen Problemen, die bei der Verarbeitung von realen Trainingsdaten durch neuronale Netze auftreten können, und die bisher in der Literatur nicht oder nicht ausreichend diskutiert wurden, wurden Lösungen präsentiert. Alle diese Verfahren wurden in einem Gesamtsystem zur Verarbeitung von Korrosionsdaten implementiert und empirisch validiert.

Ausgang aller Konzepte und Algorithmen bilden neuronale Netze mit erweiterten bayesschen Methoden: sie verarbeiten Trainingsdaten mit individuellen Messfehlerangaben. Entsprechend können zu den Prognosen auch Prognosefehler in Form von Konfidenzen berechnet werden.

Für die Implementierung wurden generalisierte lineare Netze verwendet. Sie ermöglichen einen sehr effizienten Trainingsalgorithmus, der neben den Gewichten auch die a priori Verteilung der Gewichte vollautomatisch bestimmt. Weiter wurde eine Reihe von theoretischen Aussagen präsentiert, die für das Verständnis der erweiterten bayesschen Methoden wichtig sind, und die das Verhältnis zwischen Trainings- und Prognosefehlern, den Basisfunktionen und der Gewichtsregularisierung beschreiben.

Die Kooperation von Netzen wurde eingeführt, um zwei strukturelle Probleme der vorliegenden Korrosionsdatensammlung effektiv zu lösen. Da sich die Messstellen einerseits in einem sehr hochdimensionalen Raum befinden, sie aber andererseits in vergleichsweise wenigen Clustern angeordnet sind, werden jeweils inhaltlich zusammengehörige Trainingsdaten zu einzelnen Experten zusammengefasst. Außerdem werden Trainingsdaten, die in einem Parameter fehlende, also verteilte Werte aufweisen, in anderen Experten trainiert als Trainingsdaten mit konkreten Werten. Darüber hinaus beschleunigt die Kooperation sowohl das Training als auch die Prognose und verringert den benötigten Speicherplatz.

Die Beziehung zwischen einem einzelnen Netz, das auf allen Daten trainiert wurde, und zwei kooperierenden Netzen, die zusammen auf den gleichen Daten trainiert wurden, wurde analytisch und beispielhaft untersucht. Die Kooperation generalisiert dabei näherungsweise genauso gut wie ein einzelnes, universelles Netz.

Die Korrosion ist überwiegend, aber nicht überall eine deterministische Funktion der Eingangsgrößen. Das vorgestellte Modell des regionalen Rauschens ist, wenn entsprechende Trainingsdaten zur Verfügung stehen, in der Lage, diejenigen Regionen im Eingaberaum zu erkennen, in denen Trainingsdaten, gemessen an ihren Messfehlerangaben, zueinander in Widerspruch stehen. Die Standardabweichung des inhärenten Rauschens wird dabei erkannt und bildet zusammen mit dem bayesschen Prognosefehler einen erweiterten Fehlerbalken der Prognose.

Das in der Literatur üblicherweise verwendete Klassifikationsmodell, das die Eingangsgrößen als Zufallsvariablen in Abhängigkeit der zu trainierenden Klasse annimmt, ist auf die Korrosion nicht anwendbar. Daher wurde ein alternatives Modell entwickelt, welches diese Abhängigkeit umkehrt. Es ermöglicht darüber hinaus eine Trennung der trainierten und der prognostizierten Klassen, sodass die Information, die in den Trainingsdaten enthalten ist, besser genutzt werden kann.

Die Verarbeitung von Daten, die nicht ursprünglich zum Training von neuronalen Netzen zusammengestellt wurden, erfordert eine umfangreiche Vorverarbeitung. Dazu wurden Methoden eines zweistufigen Verfahrens beschrieben, dessen zentrales Element das komplexe, benutzer- und problemorientierte konzep-

tionelle Datenschema ist. Bei der Abbildung der ursprünglichen Trainingsdaten in dieses Schema werden Spezifika der Datenbeschreibung abgebaut und so eine phänomenorientierte Beschreibung geleistet. In die weitere Abbildung auf die Netzein- und -ausgänge fließt analytisches Problemwissen ein, was dann zu erheblich verbesserten Generalisierungseigenschaften führt.

Ein Überblick über den Leistungsumfang der entstandenen Software und empirische Auswertungen, die die Leistungsfähigkeit und die Korrektheit aller beschriebenen Modelle und Konzepte belegen, schließen die Arbeit ab.

8.1.1 Wirtschaftliche Verbesserungen

Der Bayer AG konnte ein im Rahmen des Forschungsprojekts erstelltes Softwaresystem übergeben werden, das eine erhebliche Verbesserung gegenüber dem bisher verwendeten Systems darstellt.

Die Qualität der Prognosen ist entscheidend gesteigert worden. Diese begründet sich auf der einen Seite durch die Berechnung von Prognosefehlern, die ein Maß für die Konfidenz darstellen. Die zweite Seite fußt auf der Automatisierung des Trainings, das zu hoch evidenten Netzen gleichbleibender Qualität führt. Beide Eigenschaften sind für einen robusten Einsatz im betrieblichen Alltag wesentlich.

Zwar wird man sich — etwa bei der Auslegung neuer Anlagenteile — nicht allein auf Prognosen verlassen, jedoch wird sich die Zahl der Labor- und Betriebsuntersuchungen entscheidend reduzieren lassen, was wiederum zu einer Kostenreduktion führen wird. Außerdem steigt auch die Qualität der von der Abteilung Werkstofftechnik angebotenen Leistungen: wo bisher mehrwöchige Korrosionsversuche nötig waren, können jetzt in vielen Fällen fundierte Schätzungen innerhalb von Minuten bestimmt werden.

8.2 Ausblick

Wie die empirischen Ergebnisse in Kapitel 7 zeigen, funktioniert das Gesamtsystem im Wesentlichen zufriedenstellend, die Arbeit an ihm ist daher in gewisser Weise abgeschlossen. Trotzdem kann natürlich ein so umfangreiches System, das reale und damit komplex strukturierte Daten verarbeitet, an verschiedenen Stellen noch verbessert werden. Eine Reihe von kleineren Verbesserungsmöglichkeiten wurden bereits an verschiedenen Stellen kurz erwähnt. Darüber hinaus sollen hier noch einige komplexere Vorschläge bzw. Ideen vorgestellt werden, die die Leistungsfähigkeit unter Umständen stark verbessern könnten.

Die Art der Basisfunktionen der Netze wurde aufgrund einer Reihe von fundierten Argumenten getroffen. Empirische Untersuchungen dazu wurden aber nur für einen kontinuierlichen Ausgangsparameter und nur für Testdaten, die identisch den Lerndaten verteilt waren, durchgeführt. Hier sollten weitere Untersuchungen zum Extrapolationsverhalten der Netze und zu Netzen für diskontinuierliche Ausgangsparameter und für das regionale Rauschen stattfinden.

Die Anzahl der Basisfunktionen wird derzeit einfach heuristisch aus der Anzahl der Trainingsdaten bestimmt. Um aber Speicher- und Rechenzeit zu optimieren, könnte man auch die Anzahl der Basisfunktionen über die bayesschen Methoden bestimmen. Dabei könnte man entweder ein Komitee von Netzen verwenden, oder einen einzelnen, sorgfältig ausgewählten Repräsentanten aus einem solchen Komitee, entsprechend dem Verfahren bei der Gewichtsregularisierung.

Die Einteilung der Daten auf Expertenbereiche wird derzeit manuell durchgeführt. Es fehlt aktuell an konkreten Festlegungen für eine systematische Vorgehensweise. Würde diese aus theoretischen Überlegungen oder empirischen Untersuchungen heraus erarbeitet, könnte sie gegebenenfalls auch algorithmisch formuliert und implementiert werden. Diese automatische Einteilung könnte dann nicht nur erhebliche Arbeitszeit sparen, sondern würde auch eine robuste und reproduzierbare Einteilung liefern. Sollte sich bei dieser Untersuchung herausstellen, dass eine einzelne Einteilung nicht die gewünschten Eigenschaften besitzt, sollte die Einführung eines Komitees von Einteilungen (Gruppen) geprüft werden.

Das System kooperierender Netze baut auf der Disjunktheit der Datenquellen der Experten auf. Daher können sehr leicht weitere Experten eingefügt werden, die nicht auf Trainingsdaten, sondern auf explizitem Korrosionswissen aufbauen. Derartige Experten könnten etwa auf der Simulation der chemischen Abläufe bei der Korrosion basieren ([Pourbaix]). Die Anforderungen an sie beinhalten lediglich die Verarbeitung von Daten des konzeptionellen Schemas sowie die Berechnung von Prognosewerten und -fehlern. Die Verschmelzung von Experten aufbauend auf explizitem Korrosionswissen und trainierten Experten bilden Hybridmodelle. Auch ihr Einsatz könnte untersucht werden.

Anhang A

Übersicht über die verwendeten Symbole

Allgemeine Symbole

$\|.\|$	euklidische Norm (2-Norm)
COV	Kovarianz
det	Determinante (einer Matrix)
E	Erwartungswert
e_i	i-ter Einheitsvektor
$\Gamma(.)$	Gammafunktion $\Gamma(x) = \int_0^\infty t^{x-1} \exp(-t)\, dt$
I	Einheitsmatrix
$\mathbb{N}$	Menge der (positiven) natürlichen Zahlen
$\mathcal{N}(\mu, \sigma^2)$	normalverteilte Zufallsvariable mit Erwartungswert σ und Varianz σ^2
P	Wahrscheinlichkeit
p	Wahrscheinlichkeitsdichte
$\phi(.)$	Verteilungsfunktion der Standardnormalverteilung
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^+$	Menge der echt positiven reellen Zahlen
Span	Menge aller Linearkombinationen der angegebenen Vektoren
$.^T$	Transponationsoperator
tr	Spur (einer Matrix)
VAR	Varianz

Symbole der Netze

A	Hesse-Matrix der Fehlerfunktion, siehe Seite 28
A_D	Datenanteil der Hesse-Matrix, siehe Seite 35
b	Datenkonstante, siehe Seite 28
D	Menge der Trainingsdaten
$f(x)$	wahre Funktion (an der Stelle x)
$g(x, w)$	Netzfunktion (mit Gewichten w an der Stelle x)
$g(x)$	Vektor der Basisfunktionen (an der Stelle x)
L	Dimension des Eingangsraums
l	Laufindex über die Eingänge
M	Anzahl der Gewichte, Anzahl der Basisfunktionen
m	Laufindex über die Gewichte
$\mu(x)$	Prognosewert (an der Stelle x)
N	Anzahl der Trainingsdaten

n	Laufindex über die Trainingsdaten
s_n	Messfehler zum n-ten Trainingsdatensatz
σ_{w}	Standardabweichung der a priori Verteilung der Gewichte, auch: Gewichtsregularisierungsfaktor
$\sigma(x)$	Prognosefehler (an der Stelle x)
$\sigma^2(x)$	Prognosevarianz (an der Stelle x)
t_n	Messwert zum n-ten Trainingsdatensatz
w	Vektor der Gewichte
w_m	m-tes Gewicht
w_{MP}	wahrscheinlichster Gewichtsvektor
x	Eingangsvektor, Anfragestelle einer Prognose
x_n	Messstelle zum n-ten Trainingsdatensatz

Anhang B

Lemmata

Lemma 1 *Seien $h : \mathbb{R} \times \mathbb{R}^+ \to]0,1[$ eine Funktion und $t^{(1)},\ t^{(2)} \in \mathbb{R}$, $s^{(1)},\ s^{(2)} \in \mathbb{R}^+$ und $P_1,\ P_2 \in]0,1[$ vorgegebene Konstanten, die die Ungleichungen $t^{(1)} \neq t^{(2)}$ und $(s^{(1)})^2 \ln\left(\frac{P_1}{1-P_1}\right) \neq (s^{(2)})^2 \ln\left(\frac{1-P_2}{P_2}\right)$ erfüllen. An die Funktion h werden die folgenden Bedingungen gestellt:*

Bedingung I
$$h(t^{(1)}, s^{(1)}) = P_1 \tag{B.1}$$

Bedingung II
$$h(t^{(2)}, s^{(2)}) = 1 - P_2 \tag{B.2}$$

Bedingung III $\forall t_1, \ldots, t_N \in \mathbb{R},\ s_1, \ldots, s_N \in \mathbb{R}^+ :$
$$h\left(\frac{\sum_{n=1}^{N} s_n^{-2} t_n}{\sum_{n=1}^{N} s_n^{-2}}, \frac{1}{\sqrt{\sum_{n=1}^{N} s_n^{-2}}}\right) = \frac{1}{1 + \prod_{n=1}^{N}\left(\frac{1}{h(t_n, s_n)} - 1\right)} \tag{B.3}$$

Bedingung IV $h(t,s)$ *ist stetig in t und s.* (B.4)

Dann ist h durch

$$h(t,s) = \frac{1}{1 + \exp\left(-\alpha \frac{t-T}{s^2}\right)} \tag{B.5}$$

$$\alpha = \frac{c_1 - c_2}{t^{(1)} - t^{(2)}} \tag{B.6}$$

$$T = \frac{c_1 t^{(2)} - c_2 t^{(1)}}{c_1 - c_2} \tag{B.7}$$

$$c_1 = (s^{(1)})^2 \ln\left(\frac{P_1}{1-P_1}\right) \tag{B.8}$$

$$c_2 = (s^{(2)})^2 \ln\left(\frac{1-P_2}{P_2}\right) \tag{B.9}$$

gegeben und eindeutig bestimmt.

Beweis. Die Stetigkeit von h ist offensichtlich nach Gleichung B.5. Dass h die Bedingungen I, II und III erfüllt, sieht man durch einfaches Nachrechnen. Bedingung I:

$$\begin{aligned} h(t^{(1)}, s^{(1)}) &= \frac{1}{1 + \exp\left(-\alpha(t^{(1)} - T)\frac{1}{(s^{(1)})^2}\right)} \\ &= \frac{1}{1 + \exp\left(-\frac{c_1 - c_2}{t^{(1)} - t^{(2)}}\left(t^{(1)} - \frac{c_1 t^{(2)} - c_2 t^{(1)}}{c_1 - c_2}\right)\frac{1}{(s^{(1)})^2}\right)} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{1+\exp\left(-\frac{c_1-c_2}{t^{(1)}-t^{(2)}}\left(\frac{c_1t^{(1)}-c_1t^{(2)}}{c_1-c_2}\right)\frac{1}{(s^{(1)})^2}\right)}\\
&= \frac{1}{1+\exp\left(-\frac{c_1}{(s^{(1)})^2}\right)}\\
&= \frac{1}{1+\frac{1-P_1}{P_1}}\\
&= \frac{P_1}{P_1+1-P_1}\\
&= P_1. \qquad \text{(B.10)}
\end{aligned}
$$

Bedingung II kann entsprechend eingesehen werden:

$$
\begin{aligned}
h(t^{(2)},s^{(2)}) &= \frac{1}{1+\exp\left(-\alpha(t^{(2)}-T)\frac{1}{(s^{(2)})^2}\right)}\\
&= \frac{1}{1+\exp\left(-\frac{c_1-c_2}{t^{(1)}-t^{(2)}}\left(t^{(2)}-\frac{c_1t^{(2)}-c_2t^{(1)}}{c_1-c_2}\right)\frac{1}{(s^{(2)})^2}\right)}\\
&= \frac{1}{1+\exp\left(-\frac{c_1-c_2}{t^{(1)}-t^{(2)}}\left(\frac{c_2t^{(1)}-c_2t^{(2)}}{c_1-c_2}\right)\frac{1}{(s^{(2)})^2}\right)}\\
&= \frac{1}{1+\exp\left(-\frac{c_2}{(s^{(2)})^2}\right)}\\
&= \frac{1}{1+\frac{P_2}{1-P_2}}\\
&= \frac{1-P_2}{1-P_2+P_2}\\
&= 1-P_2. \qquad \text{(B.11)}
\end{aligned}
$$

Bedingung III gilt gemäß den folgenden Umformungen; man beachte, dass sie sogar für beliebige Werte von α und T gilt. Linke Seite:

$$
\begin{aligned}
h\left(\frac{\sum_{n=1}^N s_n^{-2}t_n}{\sum_{n=1}^N s_n^{-2}},\frac{1}{\sqrt{\sum_{n=1}^N s_n^{-2}}}\right) &= \frac{1}{1+\exp\left(-\alpha\left(\frac{\sum_{n=1}^N s_n^{-2}t_n}{\sum_{n=1}^N s_n^{-2}}-T\right)\sum_{n=1}^N s_n^{-2}\right)}\\
&= \frac{1}{1+\exp\left(-\alpha\left(\sum_{n=1}^N s_n^{-2}(t_n-T)\right)\right)}. \qquad \text{(B.12)}
\end{aligned}
$$

Rechte Seite:

$$
\frac{1}{1+\prod_{n=1}^N\left(\frac{1}{h(t_n,s_n)}-1\right)} = \frac{1}{1+\prod_{n=1}^N\left(\left(1+\exp\left(-\alpha\frac{t_n-T}{s_n^2}\right)\right)-1\right)}
$$

$$
\begin{aligned}
&= \frac{1}{1+\prod_{n=1}^{N}\exp\left(-\alpha\frac{t_n-T}{s_n^2}\right)} \\
&= \frac{1}{1+\exp\left(-\alpha\sum_{n=1}^{N}\frac{t_n-T}{s_n^2}\right)}.
\end{aligned} \tag{B.13}
$$

Damit wäre gezeigt, dass h alle Bedingungen erfüllt. Es bleibt die Eindeutigkeit von h zu zeigen. Setzt man für ein beliebiges $p \in \mathbb{N}$ die Variablen $t_1 := \ldots := t_p := t^{(1)}$ und $s_1 := \ldots := s_p := s^{(1)}$, dann folgt aus Bedingung III

$$
\begin{aligned}
h\left(\frac{\sum_{n=1}^{p}(s^{(1)})^{-2}t^{(1)}}{\sum_{n=1}^{p}(s^{(1)})^{-2}}, \frac{1}{\sqrt{\sum_{n=1}^{p}(s^{(1)})^{-2}}}\right) &= \frac{1}{1+\prod_{n=1}^{p}\left(\frac{1}{h(t^{(1)},s^{(1)})}-1\right)} \\
h\left(t^{(1)}, \frac{1}{\sqrt{p}}s^{(1)}\right) &= \frac{1}{1+\left(\frac{1}{P_1}-1\right)^{p}}
\end{aligned} \tag{B.14}
$$

und damit die Eindeutigkeit von h für alle Punkte $(t^{(1)}, s^{(1)}/\sqrt{p})$ mit $p \in \mathbb{N}$. Wählt man nun für ein beliebiges $q \in \mathbb{N}$ die Variablen $t_1 := \ldots := t_q := t^{(1)}$ und $s_1 := \ldots := s_q := \sqrt{q/p}\, s^{(1)}$, dann folgt aus Bedingung III

$$
\begin{aligned}
h\left(\frac{\sum_{n=1}^{q}\left(\sqrt{\frac{q}{p}}s^{(1)}\right)^{-2}t^{(1)}}{\sum_{n=1}^{q}\left(\sqrt{\frac{q}{p}}s^{(1)}\right)^{-2}}, \frac{1}{\sqrt{\sum_{n=1}^{q}\left(\sqrt{\frac{q}{p}}s^{(1)}\right)^{-2}}}\right) &= \frac{1}{1+\prod_{n=1}^{q}\left(\frac{1}{h\left(t^{(1)},\sqrt{\frac{q}{p}}s^{(1)}\right)}-1\right)} \\
h\left(t^{(1)}, \frac{1}{\sqrt{p}}s^{(1)}\right) &= \frac{1}{1+\left(\frac{1}{h\left(t^{(1)},\sqrt{\frac{q}{p}}s^{(1)}\right)}-1\right)^{q}} \\
\frac{1}{h\left(t^{(1)}, \frac{1}{\sqrt{p}}s^{(1)}\right)}-1 &= \left(\frac{1}{h\left(t^{(1)},\sqrt{\frac{q}{p}}s^{(1)}\right)}-1\right)^{q} \\
\left(\left(\frac{1}{h\left(t^{(1)}, \frac{1}{\sqrt{p}}s^{(1)}\right)}-1\right)^{\frac{1}{q}}+1\right)^{-1} &= h\left(t^{(1)}, \sqrt{\frac{q}{p}}s^{(1)}\right)
\end{aligned} \tag{B.15}
$$

und damit die Eindeutigkeit von h für alle Punkte $(t^{(1)}, \sqrt{q/p}\; s^{(1)})$ mit $p, q \in \mathbb{N}$. Da h stetig ist, folgt die Eindeutigkeit für alle Punkte $(t^{(1)}, s)$ mit $s \in \mathbb{R}^+$. Diese Überlegungen zur Eindeutigkeit kann man nun für $t^{(2)}$ und $s^{(2)}$ entsprechend anstellen und kann so die Eindeutigkeit von h für alle Punkte $(t^{(2)}, s)$ mit $s \in \mathbb{R}^+$ zeigen. Die Eindeutigkeit für alle übrigen Punkte (t, s) mit $t \in \mathbb{R} \setminus \{t^{(1)}, t^{(2)}\}$ und $s \in \mathbb{R}^+$

ergibt sich durch die Wahl

$$N := 2,\ t_1 := t^{(1)},\ t_2 := t^{(2)},\ s_1 := \sqrt{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t}}s \text{ und } s_2 := \sqrt{\frac{t^{(2)}-t^{(1)}}{t-t^{(1)}}}s \tag{B.16}$$

und Einsetzen in Bedingung III. Die linke Seite von Bedingung III ergibt sich so zu

$$\begin{aligned}
& h\left(\frac{\left(\sqrt{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t}}s\right)^{-2}t^{(1)}+\left(\sqrt{\frac{t^{(2)}-t^{(1)}}{t-t^{(1)}}}s\right)^{-2}t^{(2)}}{\left(\sqrt{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t}}s\right)^{-2}+\left(\sqrt{\frac{t^{(2)}-t^{(1)}}{t-t^{(1)}}}s\right)^{-2}}, \frac{1}{\sqrt{\left(\sqrt{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t}}s\right)^{-2}+\left(\sqrt{\frac{t^{(2)}-t^{(1)}}{t-t^{(1)}}}s\right)^{-2}}}\right) \\
= \quad & h\left(\frac{\frac{t^{(2)}-t}{t^{(2)}-t^{(1)}}s^{-2}t^{(1)}+\frac{t-t^{(1)}}{t^{(2)}-t^{(1)}}s^{-2}t^{(2)}}{\frac{t^{(2)}-t}{t^{(2)}-t^{(1)}}s^{-2}+\frac{t-t^{(1)}}{t^{(2)}-t^{(1)}}s^{-2}}, \frac{1}{\sqrt{\frac{t^{(2)}-t}{t^{(2)}-t^{(1)}}s^{-2}+\frac{t-t^{(1)}}{t^{(2)}-t^{(1)}}s^{-2}}}\right) \\
= \quad & h\left(\frac{\frac{t\cdot t^{(2)}-t\cdot t^{(1)}}{t^{(2)}-t^{(1)}}s^{-2}}{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t^{(1)}}s^{-2}}, \frac{1}{\sqrt{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t^{(1)}}s^{-2}}}\right) \\
= \quad & h(t,s)
\end{aligned} \tag{B.17}$$

während die rechte Seite

$$\frac{1}{1+\left(\frac{1}{h\left(t^{(1)},\sqrt{\frac{t^{(2)}-t^{(1)}}{t^{(2)}-t}}s\right)}-1\right)\left(\frac{1}{h\left(t^{(2)},\sqrt{\frac{t^{(2)}-t^{(1)}}{t-t^{(1)}}}s\right)}-1\right)} \tag{B.18}$$

offensichtlich eindeutig bestimmt ist. □

Lemma 2 *Seien $g_1,\ldots,g_N \in \mathbb{R}^M \setminus \{\vec{0}\}$, $s_1,\ldots,s_N \in \mathbb{R}^+$ und $\sigma_w \in \mathbb{R}^+$. Sei weiter die Matrix $A \in \mathbb{R}^{M\times M}$ durch*

$$A \quad := \quad \frac{1}{\sigma_w^2}I+\sum_{i=1}^{N}\frac{1}{s_i^2}g_i g_i^T \tag{B.19}$$

gegeben. Dann gilt für jedes $n = 1,\ldots,N$ die Ungleichung

$$g_n^T A^{-1} g_n \quad < \quad s_n^2. \tag{B.20}$$

Bemerkung. Der Zusammenhang zu generalisierten linearen Netzen nach Abschnitt 3.1 ist wie folgt: wenn $x_n \in \mathbb{R}^L$ eine Messstelle und $g : \mathbb{R}^L \to \mathbb{R}^M$ der Vektor der Basisfunktionen ist, dann gilt die Entsprechung $g_n = g(x_n)$. Zusammen mit den Messfehlern s_n $(n = 1,\ldots,N)$ und der Standardabweichung der a priori Gewichtsverteilung σ_w ergibt sich daraus die Hesse-Matrix A des Netzes. Bei einer Prognose an der Stelle x_n folgt dann eine Prognosevarianz von $\sigma^2(x_n) = g(x_n)A^{-1}g(x_n)$ und die Behauptung lautet nun $\sigma^2(x_n) < s_n^2$. Anschaulich bedeutet dies: der Prognosefehler an einer Messstelle ist stets kleiner als der zugehörige Messfehler.

Beweis. Sei $n \in \{1, \ldots, N\}$. Die Matrix A wird wie folgt zerlegt:

$$A = \underbrace{\sigma_{\mathrm{w}}^{-2} I + \sum_{i \neq n} s_i^{-2} g_i g_i^T}_{B} + s_n^{-2} g_n g_n^T. \tag{B.21}$$

Die Matrix B ist (wie A auch) positiv definit und damit invertierbar, da der Summand $\sigma_{\mathrm{w}}^{-2} I$ positiv definit und alle übrigen Summanden positiv semidefinit sind. Nun wird der folgende Ausdruck betrachtet:

$$\begin{aligned}
\frac{1}{g_n^T A^{-1} g_n} &= \frac{1}{g_n^T \left(B + s_n^{-2} g_n g_n^T\right)^{-1} g_n} \\
&= \frac{g_n^T (I + s_n^{-2} B^{-1} g_n g_n^T) g_n}{g_n^T (B + s_n^{-2} g_n g_n^T)^{-1} g_n g_n^T (I + s_n^{-2} B^{-1} g_n g_n^T) g_n} \\
&= \frac{g_n^T (I + s_n^{-2} B^{-1} g_n g_n^T) g_n}{g_n^T (B + s_n^{-2} g_n g_n^T)^{-1} (g_n g_n^T + s_n^{-2} g_n g_n^T B^{-1} g_n g_n^T) g_n} \\
&= \frac{g_n^T (I + s_n^{-2} B^{-1} g_n g_n^T) g_n}{g_n^T (B + s_n^{-2} g_n g_n^T)^{-1} (B + s_n^{-2} g_n g_n^T) B^{-1} g_n g_n^T g_n} \\
&= \frac{g_n^T (I + s_n^{-2} B^{-1} g_n g_n^T) g_n}{g_n^T B^{-1} g_n g_n^T g_n} \\
&= \frac{g_n^T g_n + s_n^{-2} g_n^T B^{-1} g_n g_n^T g_n}{g_n^T B^{-1} g_n g_n^T g_n} \\
&= \frac{1}{g_n^T B^{-1} g_n} + s_n^{-2} \\
&> s_n^{-2}.
\end{aligned} \tag{B.22}$$

Durch Kehrwertbildung ergibt sich die Behauptung. □

Lemma 3 *Seien $s_1, \ldots, s_N \in \mathbb{R}^+$. Seien weiter die folgenden Variablen definiert:*

$$\sigma := \left(\sum_{i=1}^{N} s_i^{-2} \right)^{-\frac{1}{2}} \tag{B.23}$$

$$z_n := \frac{1}{1 - \sigma^2 s_n^{-2}} \left(1 - 2\sigma^2 s_n^{-2} + \sigma^4 \sum_{i=1}^{N} s_i^{-4} \right) \qquad \textit{für} \qquad n = 1, \ldots, N. \tag{B.24}$$

Dann gelten folgende scharfe Abschätzungen:

$$1 \leq \sum_{n=1}^{N} \frac{1}{N} z_n < 2 \tag{B.25}$$

$$0 < \sum_{n=1}^{N} \frac{s_n^{-k}}{\sum_{i=1}^{N} s_i^{-k}} z_n \leq 1 \qquad \textit{für} \qquad k \geq 2. \tag{B.26}$$

Beweis. Jede der vier Ungleichungen wird hier einzeln bewiesen. Zunächst wird dazu der mittlere Term der Ungleichungen B.25 umgeformt:

$$\sum_{n=1}^{N} \frac{1}{N} z_n = \frac{1}{N} \sum_{n=1}^{N} \left(1 + \frac{-\sigma^2 s_n^{-2} + \sigma^4 \sum_{i=1}^{N} s_i^{-4}}{1 - \sigma^2 s_n^{-2}} \right)$$

$$
\begin{aligned}
&= 1 + \frac{1}{N}\sum_{n=1}^{N} \frac{-\sigma^2 s_n^{-2} + \sigma^4 \sum_{i=1}^{N} s_i^{-4}}{1 - \sigma^2 s_n^{-2}} \\
&= 1 + \frac{1}{N}\sum_{n=1}^{N} \sigma^2 \frac{-\sigma^{-2} s_n^{-2} + \sum_{i=1}^{N} s_i^{-4}}{\sigma^{-2} - s_n^{-2}} \\
&= 1 + \frac{1}{N}\sum_{n=1}^{N} \sigma^2 \frac{-\sum_{i=1}^{N} s_i^{-2} s_n^{-2} + \sum_{i=1}^{N} s_i^{-4}}{\sum_{i=1}^{N} s_i^{-2} - s_n^{-2}} \\
&= 1 + \frac{1}{N}\sum_{n=1}^{N} \sigma^2 \frac{-\sum_{i\neq n} s_i^{-2} s_n^{-2} + \sum_{i\neq n} s_i^{-4}}{\sum_{i\neq n} s_i^{-2}} \\
&= 1 + \frac{1}{N}\sum_{n=1}^{N} \sigma^2 \left(\frac{\sum_{i\neq n} s_i^{-4}}{\sum_{i\neq n} s_i^{-2}} - s_n^{-2} \right). \qquad \text{(B.27)}
\end{aligned}
$$

Der Zähler des Bruchs in der Klammer wird nun nach oben und unten abgeschätzt. Es gilt

$$
\begin{aligned}
\left(\sum_{i\neq n} s_i^{-2}\right)^2 &= \sum_{i\neq n}\sum_{j\neq n} s_i^{-2} s_j^{-2} \\
&= \sum_{i\neq n}\sum_{j\neq n} s_i^{-2} s_j^{-2} - \frac{1}{2} s_i^{-4} - \frac{1}{2} s_j^{-4} + \frac{1}{2}(s_i^{-4} + s_j^{-4}) \\
&= \sum_{i\neq n}\sum_{j\neq n} -\frac{1}{2}(s_i^{-2} - s_j^{-2})^2 + \frac{1}{2}(s_i^{-4} + s_j^{-4}) \\
&\leq \sum_{i\neq n}\sum_{j\neq n} \frac{1}{2}(s_i^{-4} + s_j^{-4}) \\
&= \frac{1}{2}(N-1)\sum_{i\neq n} s_i^{-4} + \frac{1}{2}(N-1)\sum_{j\neq n} s_j^{-4} \\
&= (N-1)\sum_{i\neq n} s_i^{-4}. \qquad \text{(B.28)}
\end{aligned}
$$

Daraus lässt sich nun Ausdruck B.27 abschätzen:

$$
\begin{aligned}
\sum_{n=1}^{N} \frac{1}{N} z_n &\geq 1 + \frac{1}{N}\sum_{n=1}^{N} \sigma^2 \left(\frac{\sum_{i\neq n} s_i^{-2}}{N-1} - s_n^{-2} \right) \\
&= 1 + \frac{1}{N}\sum_{n=1}^{N} \sigma^2 \left(\frac{\sigma^{-2} - s_n^{-2}}{N-1} - s_n^{-2} \right) \\
&= 1 + \frac{1}{N}\sigma^2 \left(\frac{N\sigma^{-2} - \sigma^{-2}}{N-1} - \sigma^{-2} \right) \\
&= 1, \qquad \text{(B.29)}
\end{aligned}
$$

womit die linke Ungleichung der Behauptung B.25 bewiesen wäre. Eine obere Schranke des Zählers aus Gleichung B.27 erhält man durch folgende Betrachtung:

$$\begin{aligned}\left(\sum_{i\neq n} s_i^{-2}\right)^2 &= \sum_{i\neq n}\sum_{j\neq n} s_i^{-2}s_j^{-2} \\ &= \sum_{i\neq n} s_i^{-4} + \sum_{i\neq n}\sum_{\substack{j\neq n,\\ j\neq i}} s_i^{-2}s_j^{-2} \\ &\geq \sum_{i\neq n} s_i^{-4}. \end{aligned} \tag{B.30}$$

Setzt man dies in Gleichung B.27 ein, erhält man

$$\begin{aligned}\sum_{n=1}^{N}\frac{1}{N}z_n &\leq 1+\frac{1}{N}\sum_{n=1}^{N}\sigma^2\left(\sum_{i\neq n} s_i^{-2} - s_n^{-2}\right) \\ &= 1+\frac{1}{N}\sum_{n=1}^{N}\sigma^2\left(\sigma^{-2} - 2s_n^{-2}\right) \\ &= 1+\frac{1}{N}\sigma^2\left(N\sigma^{-2} - 2\sigma^{-2}\right) \\ &= 1+\frac{N-2}{N}, \end{aligned} \tag{B.31}$$

woraus die behauptete rechte Ungleichung B.25 folgt. Die linke Ungleichung B.26 kann wie folgt eingesehen werden:

$$\begin{aligned}\sum_{n=1}^{N}\frac{s_n^{-k}}{\sum_{i=1}^{N} s_i^{-k}}z_n &= \sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\left(1-\sigma^2 s_n^{-2}\right)}\left(1-2\sigma^2 s_n^{-2}+\sigma^4\sum_{i=1}^{N} s_i^{-4}\right) \\ &\geq \sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\left(1-\sigma^2 s_n^{-2}\right)}\left(1-2\sigma^2 s_n^{-2}+\sigma^4 s_n^{-4}\right) \\ &= \sum_{n=1}^{N}\frac{s_n^{-k}(1-\sigma^2 s_n^{-2})}{\sum_{i=1}^{N} s_i^{-k}} \\ &= \sum_{n=1}^{N}\frac{s_n^{-k}(\sigma^{-2}-s_n^{-2})}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\sigma^{-2}} \\ &= \sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\sigma^{-2}}\sum_{i\neq n} s_i^{-2} \\ &> 0, \end{aligned} \tag{B.32}$$

da jeder Summand positiv ist. Die rechte Ungleichung B.26 ergibt sich durch folgende Umformungen:

$$\begin{aligned}\sum_{n=1}^{N}\frac{s_n^{-k}}{\sum_{i=1}^{N} s_i^{-k}}z_n &= \sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\left(1-\sigma^2 s_n^{-2}\right)}\left((1-\sigma^2 s_n^{-2})-\sigma^2 s_n^{-2}+\sigma^4\sum_{i=1}^{N} s_i^{-4}\right) \\ &= \sum_{n=1}^{N}\frac{s_n^{-k}}{\sum_{i=1}^{N} s_i^{-k}} + \sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\left(1-\sigma^2 s_n^{-2}\right)}\left(-\sigma^2 s_n^{-2}+\sigma^4\sum_{i=1}^{N} s_i^{-4}\right) \\ &= 1+\sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N} s_i^{-k}\right)\sigma^{-4}(1-\sigma^2 s_n^{-2})}\left(-\sigma^{-2}s_n^{-2}+\sum_{i=1}^{N} s_i^{-4}\right) \end{aligned}$$

$$
\begin{aligned}
&= 1+\sum_{n=1}^{N}\frac{s_n^{-k}}{\left(\sum_{i=1}^{N}s_i^{-k}\right)\sigma^{-4}(1-\sigma^2 s_n^{-2})}\left(-\sum_{i=1}^{N}s_i^{-2}s_n^{-2}+\sum_{i=1}^{N}s_i^{-4}\right)\\
&= 1+\sum_{n=1}^{N}\sum_{i=1}^{N}\frac{s_i^{-2}s_n^{-2}}{\left(\sum_{i=1}^{N}s_i^{-k}\right)\sigma^{-4}}\cdot\frac{(s_i^{-2}-s_n^{-2})s_n^{2-k}}{1-\sigma^2 s_n^{-2}}\\
&= 1+\frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{N}\frac{s_i^{-2}s_n^{-2}}{\left(\sum_{i=1}^{N}s_i^{-k}\right)\sigma^{-4}}\left(\frac{(s_i^{-2}-s_n^{-2})s_n^{2-k}}{1-\sigma^2 s_n^{-2}}+\frac{(s_n^{-2}-s_i^{-2})s_i^{2-k}}{1-\sigma^2 s_i^{-2}}\right)\\
&= 1+\frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{N}\frac{s_i^{-2}s_n^{-2}}{\left(\sum_{i=1}^{N}s_i^{-k}\right)\sigma^{-4}}(s_i^{-2}-s_n^{-2})\left(\frac{s_n^{2-k}}{1-\sigma^2 s_n^{-2}}-\frac{s_i^{2-k}}{1-\sigma^2 s_i^{-2}}\right). \qquad \text{(B.33)}
\end{aligned}
$$

Es wird nun bewiesen, dass jeder Summand der Doppelsumme über n und i nicht positiv ist. Dies geht so: falls $s_i = s_n$ ist, verschwindet der mittlere Faktor. Aufgrund der Symmetrie der Summanden in i und n sei o.B.d.A. $s_i > s_n$, dann folgt:

$$
\begin{aligned}
s_i^{-2} &< s_n^{-2}\\
\sigma^2 s_i^{-2} &< \sigma^2 s_n^{-2}\\
1-\sigma^2 s_i^{-2} &> 1-\sigma^2 s_n^{-2}\\
\frac{1}{1-\sigma^2 s_i^{-2}} &< \frac{1}{1-\sigma^2 s_n^{-2}}\\
\frac{s_i^{2-k}}{1-\sigma^2 s_i^{-2}} &< \frac{s_n^{2-k}}{1-\sigma^2 s_n^{-2}}. \qquad \text{(B.34)}
\end{aligned}
$$

Somit sind die Summanden in B.33 nicht positiv, da der erste Faktor positiv und die beiden übrigen Faktoren entgegengesetztes Vorzeichen haben. Damit ist dann die rechte Ungleichung B.26 bewiesen.

Es bleibt noch der Beweis der Schärfe aller vier Ungleichungen. Dazu sei zunächst $s_1 := \ldots := s_N := 1$ gewählt. Es folgt aus den Definitionen

$$
\begin{aligned}
\sigma &= \left(\sum_{i=1}^{N}1\right)^{-\frac{1}{2}}\\
&= N^{-1/2} \qquad \text{(B.35)}\\
z_n &= \frac{1}{1-N^{-1}}\left(1-2N^{-1}+N^{-2}\sum_{i=1}^{N}1\right)\\
&= \frac{1}{1-N^{-1}}\left(1-2N^{-1}+N^{-1}\right)\\
&= 1 \qquad \text{(B.36)}
\end{aligned}
$$

und daraus die Schärfe der Ungleichung B.25 links

$$
\sum_{n=1}^{N}\frac{1}{N}z_n = 1 \qquad \text{(B.37)}
$$

sowie die Schärfe der Ungleichung B.26 rechts

$$
\begin{aligned}
\sum_{n=1}^{N}\frac{s_n^{-k}}{\sum_{i=1}^{N}s_i^{-k}}z_n &= \frac{\sigma^{-2}}{\sigma^{-2}}\\
&= 1. \qquad \text{(B.38)}
\end{aligned}
$$

Die anderen beiden Ungleichungen werden scharf für die Wahl $s_1 \to 0$ und $s_2 := \ldots := s_N := 1$. Es folgt

$$\begin{aligned}
\sigma &= \left(s_1^{-2} + N - 1\right)^{-\frac{1}{2}} && \text{(B.39)}\\
z_n &= \frac{1 - 2(s_1^{-2} + N - 1)^{-1} s_n^{-2} + (s_1^{-2} + N - 1)^{-2}(s_1^{-4} + N - 1)}{1 - (s_1^{-2} + N - 1)^{-1} s_n^{-2}} \\
&= \frac{(s_1^{-2} + N - 1)^2 - 2(s_1^{-2} + N - 1)s_n^{-2} + s_1^{-4} + N - 1}{(s_1^{-2} + N - 1)(s_1^{-2} + N - 1 - s_n^{-2})} && \text{(B.40)}\\
z_1 &= \frac{(s_1^{-2} + N - 1)^2 - 2(s_1^{-2} + N - 1)s_1^{-2} + s_1^{-4} + N - 1}{(s_1^{-2} + N - 1)(s_1^{-2} + N - 1 - s_1^{-2})} \\
&= \frac{s_1^{-4} + 2(N-1)s_1^{-2} + (N-1)^2 - 2s_1^{-4} - 2(N-1)s_1^{-2} + s_1^{-4} + N - 1}{(s_1^{-2} + N - 1)(N-1)} \\
&= \frac{(N-1)^2 + N - 1}{(s_1^{-2} + N - 1)(N-1)} \\
&= \frac{N}{s_1^{-2} + N - 1} && \text{(B.41)}\\
z_2 = \ldots = z_N &= \frac{(s_1^{-2} + N - 1)^2 - 2(s_1^{-2} + N - 1) + s_1^{-4} + N - 1}{(s_1^{-2} + N - 1)(s_1^{-2} + N - 1 - 1)} \\
&= \frac{s_1^{-4} + 2(N-1)s_1^{-2} + (N-1)^2 - 2s_1^{-2} - 2(N-1) + s_1^{-4} + N - 1}{(s_1^{-2} + N - 1)(s_1^{-2} + N - 2)} \\
&= \frac{2s_1^{-4} + 2(N-2)s_1^{-2} + (N-1)^2 - (N-1)}{(s_1^{-2} + N - 1)(s_1^{-2} + N - 2)} \\
& && \text{(B.42)}
\end{aligned}$$

und für den mittleren Term aus den Ungleichungen B.25:

$$\begin{aligned}
\lim_{s_1 \to 0} z_1 &= 0 && \text{(B.43)}\\
\lim_{s_1 \to 0} z_2 &= \lim_{s_1 \to 0} \frac{2s_1^{-4}}{s_1^{-2} s_1^{-2}} \\
&= 2 && \text{(B.44)}\\
\lim_{s_1 \to 0} \sum_{n=1}^{N} \frac{1}{N} z_n &= \lim_{s1 \to 0} \frac{1}{N} \left(z_1 + (N-1) z_2\right) \\
&= 2\frac{N-1}{N}, && \text{(B.45)}
\end{aligned}$$

was für $N \to \infty$ die Schärfe der rechten Ungleichung B.25 zeigt. Der mittlere Term der Ungleichungen B.26 bildet unter der gleichen Wahl der Zahlen $s_1, \ldots, s_N$ die Formen

$$\begin{aligned}
\lim_{s_1 \to 0} \frac{s_1^{-k}}{\sum_{i=1}^{N} s_i^{-k}} z_1 &= \lim_{s_1 \to 0} \frac{s_1^{-k}}{s_1^{-k} + N - 1} z_1 \\
&= 1 \cdot 0 && \text{(B.46)}\\
\lim_{s_1 \to 0} \frac{s_2^{-k}}{\sum_{i=1}^{N} s_i^{-k}} z_2 &= \lim_{s_1 \to 0} \frac{1}{s_1^{-k} + N - 1} z_2 \\
&= 0 \cdot 2 && \text{(B.47)}\\
\lim_{s_1 \to 0} \sum_{n=1}^{N} \frac{s_n^{-k}}{\sum_{i=1}^{N} s_i^{-k}} z_n &= \lim_{s_1 \to 0} \left(\frac{s_1^{-k}}{\sum_{i=1}^{N} s_i^{-k}} z_1 + (N-1) \frac{s_2^{-k}}{\sum_{i=1}^{N} s_i^{-k}} z_2 \right) \\
&= 0, && \text{(B.48)}
\end{aligned}$$

womit die Schärfe der linken Ungleichung B.26 bewiesen ist. □

Literaturverzeichnis

[AlkKit] F. M. Alkoot, J. Kittler, *Experimental evaluation of expert fusion strategies.* Pattern Recognition Letters 20, 1361–1369, 1999

[AmaMur] S. Amari, N. Murata, *Statistical Analysis of Regularization Constant — From Bayes, MDL and NIC Points of View.* IWANN 97, 284–293, 1997

[Azizi] J. Azizi, *Modellierung und Qualität von Korrosionsdaten aus Faktendatenbanken.* Diplomarbeit, Uni Bonn, Institut für Informatik II, November 2001

[AvnInt] R. Avnimelech, N. Intrator, *Boosted Mixture of Experts: An Ensemble Learning Scheme.* Neural Computation 11, 483–497, 1999

[BaeBie] F. Bärmann, F. Biegler-König, *On a class of efficient learning algorithms for neural networks.* Neural Networks 5, 139–144, 1992

[Barron] A. R. Barron, *Universal Approximation Bounds for Superpositions of a Sigmoidal Function.* IEEE Transactions on Information Theory 39/3, 930–945, 1993

[Battiti] R. Battiti, *Using Mutual Information for Selecting Features in Supervised Neural Net Learning.* IEEE Transactions on Neural Networks 5, 537–550, 1994

[Bayes] T. Bayes, *An essay toward solving a problem in the doctrine of chances.* Philosophical Transactions of the Royal Society of London 53, 370–418, 1764. Reprinted in: E. S. Pearson, M. G. Kendall (eds.), *Studies in the History of Statistics and Probability* Charles Griffin, London, 131–153, 1970.

[Berger] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, New York, second edition, 1985, Kapitel 3

[BerSch] Bergmann, Schaefer, *Lehrbuch der Experimentalphysik. Band 6: Festkörper.* de Gruyter, Berlin, 1992, Kapitel 5

[Bidasaria] H. B. Bidasaria, *Least desirable feature elimination in a general pattern recognition problem.* Pattern Recognition 20, 365–370, 1987

[BioMeePot] J. C. Bioch, O. van der Meer, R. Potharst, *Classification using Bayesian Neural Nets.* Technical Report eur-cs-95-09, Erasmus University Rotterdam, 1996

[Bishop] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995

[BleOba] S. A. Bleha, M. S. Obaidat, *Dimensionality Reduction and Feature Extraction Applications in Identifying Computer Users.* IEEE transactions on systems, man and cybernetics 21, 452–456, 1991

[Breiman] L. Breiman, *Bagging Predictors.* Machine Learning 26, 123–140, 1996

[BulHoo] A. Bulsari, P. Hooli, *More accurate alloying with neural networks.* Stainless Steel World, 54–57, 2000

[Campbell] W. M. Campbell, *Generalized linear discriminant sequence kernels for speaker recognition.* Motorola Human Interface Lab, Tempe, 2002, http://citeseer.nj.nec.com/483492.html

[CarCunBha] J. G. Carney, P. Cunningham, U. Bhagwan, *Confidence and prediction intervals for neural network ensembles.* Proceedings of IJCNN'99, The International Joint Conference on Neural Networks, Washington, 1999

[CheAnd] C. K. Chen, H. C. Andrews, *Nonlinear Intrinsic Dimensionality Computations.* IEEE transactions on computers, 178–183, 1974

[CibSouGal] T. Cibas, F. F. Soulie, P. Gallinari, S. Raudys, *Variable selection with neural networks.* Neurocomputing, 223–248, 1996

[CotQinOwe] R. A. Cottis, L. Qing, G. Owen, S. J. Gartland, I. A. Helliwell, M. Turega, *Neural network methods for corrosion data reduction.* Materials and Design 20, Elsevier, 169–178, 1999

[DIN50900] Normelausschuss Materialprüfung (NMP) im DIN, *DIN 50 900, Teil 2: Korrosion der Metalle, Elektrochemische Begriffe.* Deutsches Institut für Normung e. V., Beuth Verlag, Berlin, 1984

[DIN50918] Normelausschuss Materialprüfung (NMP) im DIN, *DIN 50 918: Korrosion der Metalle, Elektrochemische Korrosionsuntersuchungen.* Deutsches Institut für Normung e. V., Beuth Verlag, Berlin, 1978

[DybRob] R. Dybowski, S. Roberts, *Confidence intervals and prediction intervals for feed-forward neural networks.* In: R. Dybowski, V. Gant (eds.), *Clinical Applications of Artificial Neural Networks.* Cambridge University Press, 298–326, 2001

[Forster] O. Forster, *Analysis 1. Differential- und Integralrechnung einer Veränderlichen.* Vieweg, Braunschweig, 4. Auflage, 1983

[Forster] O. Forster, *Analysis 3. Integralrechnung im $\mathbb{R}^n$ mit Anwendungen.* Vieweg, Braunschweig, 3. Auflage, 1984

[FoxCawTal] R. J. Foxall, G. C. Cawley, N. L. C. Talbot, S. R. Dorling, D. P. Mandic, *Heteroscedastic Regularised Kernel Regression for Prediction of Episodes of Poor Air Quality.* Proc. of European Symposium on Artificial Neural Networks (ESANN), 19–24, 2002

[FriFinWai] J. Fritsch, M. Finke, A. Waibel, *Adaptively Growing Hierarchical Mixtures of Experts.* In: M. C. Mozer, M. I. Jordan, T. Petsche (eds.), *Advances in Neural Information Processing Systems 9.* MIT Press, 459–465, 1997

[Gräfen] H. Gräfen, *VDI Lexikon: Werkstofftechnik.* ISBN 3-18-401328-6, VDI Verlag, 538–543, 1993

[GuyMatVap] I. Guyon, N. Matic, V. Vapnik, *Discovering Informative Patterns and Data Cleaning.* In: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining.* AAAI Press/The MIT Press, Menlo Park, California, 181-203, 1996

[Heskes] T. Heskes, *Practical confidence and prediction intervals.* In: M. Mozer, M. Jordan, T.Petsche (eds.), *Advances in Neural Information Processing Systems 9*, MIT Press, Cambridge, 176-182, 1997

[ImpSal] S. Impedovo, A. Salzo, *A new evaluation method for expert combination in multi-expert system designing.* Proceedings of Multiple Classifier Systems (MCS 2000). In: J. Kittler, F. Roli (eds.), *Lecture Notes in Computer Science 1857*, Springer-Verlag, Berlin, 230–239, 2000

[IshMiyTan] H. Ishibuchi, A. Miyazaki, H. Tanaka, *Neural-network-based diagnosis systems for incomplete data with missing inputs.* Proceedings of the IEEE International Conference on Neural Networks 6, 3457-3460, 1994

[JacJorNow] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, *Adaptive Mixtures of Local Experts.* Neural Computation 3, 79–87, 1991

[JacTanPen] R. A. Jacobs, M. A. Tanner, F. Peng, *Bayesian inference for hierarchical mixtures-of-experts with applications to regression and classification.* Statistical Methods in Medical Research 5, 375–390, 1996

[JiaTan1] sc W. Jiang, M. A. Tanner, *On the Approximation Rate of Hierarchical Mixtures-of-Experts for Generalized Linear Models.* Neural Computation 11, 1183–1198, 1999

[JiaTan2] sc W. Jiang, M. A. Tanner, *On the identifiability of mixtures-of-experts.* Neural Networks 12, 1253–1258, 1999

[KatKat] S. Katz, A. S. Katz, *Supervised neural networks capable of training on censored data and incomplete patterns.* World Congress on Neural Networks, 1221–1226, 1996

[KonDie] E. B. Kong, T. G. Dietterich, *Error-Correcting Output Coding Corrects Bias and Variance.* Proceedings of the International Conference on Machine Learning, 313-321, 1995

[Kulikowski] C. A. Kulikowski, *Discriminatory Dimensionality Reduction.* IEEE transactions on information theory, 498-499, 1971

[Kuncheva] L. I. Kuncheva, *A theoretical study on expert fusion strategies.* IEEE Transactions on Pattern Analysis and Machine Intelligence, submitted 2000, http://citeseer.nj.nec.com/kuncheva00theoretical.html

[LamVeh] J. Lampinen, A. Vehtari, *Bayesian Techniques for Neural Networks — Review and Case Studies.* In: M. Gabbouj, P. Kuosmanen (eds.), *Proceedings of Eusipco, X. European Signal Processing Conference*, Tampere, Finland, 713–720, 2000

[MacKay1] D. J. C. MacKay, *Bayesian Interpolation.* Neural Computation 4, 415–447, 1992

[MacKay2] D. J. C. MacKay, *A Practical Bayesian Framework for Backpropagation Networks.* Neural Computation 4, 448–472, 1992

[MacKay3] D. J. C. MacKay, *The Evidence Framework Applied to Classification Networks.* Neural Computation 4, 720–736, 1992

[MacKay4] D. J. C. MacKay, *Bayesian Non-Linear Modelling with Neural Networks.* http://www.inference.phy.cam.ac.uk/mackay/BayesNets.html

[Meister] A. Meister, *Numerik linearer Gleichungssysteme. Eine Einführung in moderne Verfahren.* Vieweg Verlag, 1999

[MixJon] D. F. Mix, R. A. Jones, *A Dimensionality Reduction Technique Based on a Least Squared Error Criterion.* IEEE transactions on pattern analysis and machine intelligence 4, 537–544, 1982

[Möbius] D. Möbius, *Kooperierende Neuronale Netze zur Bearbeitung von Korrosionsproblemen.* Diplomarbeit, Uni Bonn, Institut für Informatik II, Februar 1999

[Moerland] P. Moerland, *Classification using localized mixtures of experts.* Proceedings of the International Conference on Artificial Neural Networks, 838–843, 1999

[MrzLoo] T. Mrziglod, R. Loosen, Private Kommunikation mit Mitgliedern der Abteilung „Mathematische Methoden und Modelle“ der Bayer AG, 2002.

[MülIns] P. Müller, D. S. Insua, *Issues in Bayesian analysis of neural network models.* Neural Computation 10, 571–592, 1998

[Müller] P. H. Müller, *Lexikon der Stochastik.* Akademie-Verlag, Berlin, 5. Auflage, 1991

[NieWer] H. Niemeyer, E. Wermuth, *Lineare Algebra. Analytische und numerische Behandlung.* Vieweg, Braunschweig, 1987

[NixWei] D. .A. Nix, A. S. Weigend, *Learning Local Error Bars for Nonlinear Regression.* In: G. Tesauro, D. S. Touretzky, T. K. Leen (eds.), *Advances in Neural Information Processing Systems 7. (NIPS 94)* MIT Press, Cambridge, 489–496, 1995

[PenJacTan] F. Peng, R. A. Jacobs, M. A. Tanner, *Bayesian Inference in Mixtures-of-Experts and Hierarchical Mixtures-of-experts Models With an Application to Speech Recognition.* Journal of the American Statistical Association 91, 953–960, 1996

[PenRob] W. D. Penny, S. J. Roberts, *Bayesian neural networks for classification: How useful is the evidence framework ?* Neural Networks 12, 877-892, 1999

[PerCoo] M. P. Perrone, L. N. Cooper, *When networks disagree: ensemble methods for hybrid neural networks.* In: R. J. Mammone (ed.), *Artificial Neural Networks for Speech and Vision* Chapman & Hall, London, 126–142, 1993

[PosMar] W. L. Poston, D. J. Marchette, *Recursive dimensionality reduction using Fisher's linear discriminant.* Pattern Recognition 31, 881-888, 1998

[Pourbaix] M. Pourbaix, *Atlas of Electrochemical Equilibria in Aqueous Solutions.* Publ. NACE and CEBELCOR, 1966

[PreTeuVet] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C++.* Cambridge University Press, Cambridge, second edition, 2002

[QazWilBis] C. S. Qazaz, C. K. I. Williams, C. M. Bishop, *An Upper Bound on the Bayesian Error Bars for Generalized Linear Regression.* Technical report NCRG/96/005, Neural Computing Research Group, Aston University, 1996, http://www.ncrg.aston.ac.uk/cgi-bin/tr_avail.pl?trnumber=NCRG/96/005

[RafWil] M. Y. Rafiq, C. Williams, *An investigation into the integration of neural networks with the structured genetic algorithm to aid conceptual design.* In: I. Smith (ed.), *Artificial Intelligence in Structural Engineering. Information Technology for Design, Collaboration, Maintenance, and Monitoring.* Springer-Verlag, Berlin, 295-307, 1998

[RaoMilRos] A. V. Rao, D. Miller, K. Rose, A. Gersho, *Mixture of Experts Regression Modeling by Deterministic Annealing.* IEEE transactions on signal processing 45, 2811–2819, 1997

[Ripley] B. D. Ripley, *Statistical Ideas for Selecting Network Architectures.* In: B. Knappen, S. Gielen (eds.), *Neural Networks: Artificial Intelligence and Industrial Applications.* Springer, London, 183–190, 1995

[Sarle] W. S. Sarle, *Stopped Training and Other Remedies for Overfitting.* Proceedings of the 27th Symposion on the Interface, 1995

[SchBroRee] J. M. Schooling, M. Brown, P. A. S. Reed, *An example of the use of neural computing techniques in material science — the modelling of fatigue thresholds in Ni-base superalloys.* Materials Science and Engineering A260, 222–239, 1999

[Steinmeier] B. Steinmeier, *Konzeption und Implementierung eines datenbankgestützten JAVA-Programms zur Auswahl und Anzeige von Korrosionsinformationen.* Diplomarbeit, FH Osnabrück, FB E-Technik und Informatik, Juli 2000

[SykDorRap] P. Sykacek, G. Dorffner, P. Rappelsberger, J. Zeitlhofer, *Evaluating confidence measures in a neural network based sleep stager.* IEEE Trans. Biomed. Engineering, NIPS-97, Austrian Research Institute for Artificial Intelligence, Vienna, Technical Report TR-97-21, 1997

[Thodberg] H. H. Thodberg, *A Review of Bayesian Neural Networks with an Application to Near Infrared Spectroscopy.* IEEE transactions on neural networks 7, 56–72, 1996

[TitLik] M. K. Titsias, A. C. Likas, *Shared Kernel Models for Class Conditional Density Estimation.* IEEE transactions on neural networks 12, 987–997, 2001

[Torrieri] D. Torrieri, *The eigenspace separation transform for neural-network classifier.* Neural Networks 12, 419–427, 1999

[UtsWei] W. Utschick, W. Weichselberger, *Stochastic Organization of Output Codes in Multiclass Learning Problems.* Neural Computation 13, 1065–1102, 2001

[Vieten] D. Vieten, *Bayessche Methoden für neuronale Netze zur Anwendung auf Korrosionsdaten.* Diplomarbeit, Uni Bonn, Institut für Informatik II, November 2001

[Vossen] G. Vossen, *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme.* Oldenbourg Wissenschaftsverlag, München, 4. Auflage, 2000

[WatMacRob] S. Waterhouse, D. MacKay, T. Robinson, *Bayesian Methods for Mixtures of Experts.* In: D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems.* The MIT Press, 351–357, 1996

[Weber] K. E. Weber, *An Alternative Model for Classification by Neural Networks based on Bayesian Methods.* Submitted to Neurocomputing in July 2002

[WebSchSch] K. E. Weber, W. Schlagner, K. Schweier, *Estimating Regional Noise on Neural Network Predictions.* Submitted to Pattern Recognition in July 2002, accepted in February 2003, scheduled to issue 36:10

[WeiNix] A. S. Weigend, D. A. Nix, *Predictions with Confidence Intervals (Local Error Bars).* Proceedings of the International Conference on Neural Information Processing (ICONIP'94), Seoul, 847–852, 1994

[Wendler1] R. Wendler, *Optimierung des Systems kooperierender Neuronaler Netze zur Bearbeitung von Korrosionsproblemen.* Praxissemesterbericht, FH Osnabrück, FB E-Technik und Informatik, WS 2001/02

[Wendler2] R. Wendler, *Entwurf und Implementierung eines Optimierungsalgorithmus über Prognosefunktionen von neuronalen Netzen.* Diplomarbeit, FH Osnabrück, FB E-Technik und Informatik, Juli 2002

[WilQazBis] C. K. I. Williams, C. Qazaz, C. M. Bishop, H. Zhu, *On the relationship between Bayesian error bars and the input data density.* Fourth International Conference on Artificial Neural Networks, University of Cambrigde, IEE Conference Publication 409, 160–165, 1995

[Williams] P. M. Williams, *Bayesian Regularization and Pruning Using a Laplace Prior.* Neural Computation 7, 117–143, 1994

[XuJorHin] L. Xu, M. I. Jordan, G. E. Hinton, *A Modified gating network for the mixtures of experts architecture.* Proceedings of WCNN'94, San Diego, Volume 2, 405-410, 1994

[Xu] L. Xu, *RBF nets, mixture experts, and Bayesian Ying-Yang learning.* Neurocomputing 19, 223–257, 1998

[Zell] A. Zell, *Simulation neuronaler Netze.* Oldenbourg Verlag, München, 2. Nachdruck, 1997

[ZhuRoh] H. Zhu, R. Rohwer, *Bayesian Regression Filters and the Issue of Priors.* Neural Computation and Application 4, 130–142, 1996

Lebenslauf

Name	Karsten Ernst Weber
Anschrift	Höhestraße 36, 51399 Burscheid
geboren	12.7.1972 in Leverkusen
Vater	Gert Weber, Diplomkaufmann
Mutter	Jutta Weber, geb. Müller, Lehrerin
79–83	Montanus-Grundschule in Burscheid
83–92	Werner-Heisenberg-Gymnasium in Leverkusen
6/92	Abitur mit Note 1,0
7/92–9/93	Zivildienst beim Hausnotrufdienst des Caritasverbands e.V., Leverkusen
WS 93/94	Beginn des Studiums der Informatik an der Universität Bonn
10/95	Vordiplom mit Note „sehr gut“
8/96–11/97	Anstellung bei der interactive instruments GmbH, Bonn, neben dem Studium. Mitentwicklung an einem objektorientierten DBMS.
WS 98/99	Studentische Hilfskraft an der Universität Bonn. Übungsleiter der Vorlesung „Technische Informatik I“ von Prof. Anlauf.
8/99	Diplom mit Note „ausgezeichnet“
10/99–2/00	Anstellung bei der Bayer AG, Leverkusen, als Praktikant. Einarbeitung in die vorhandene Software als Vorbereitung auf die Promotion.
2/00–12/02	Wissenschaftlicher Mitarbeiter (halbe Stelle) der Universität Bonn bei Prof. Anlauf. Durchführung des Drittmittelprojekts „Optimierung des Systems kooperierender neuronaler Netze für Korrosionsprobleme“ in Kooperation mit der Bayer AG.