FORWISS
BAYERISCHES FORSCHUNGSZENTRUM
FüR WISSENSBASIERTE SYSTEME
Bereich Neuronale Netze und Fuzzy Logik

AENEAS

Anwendung und Entwicklung Neuronaler Verfahren zur Autonomen Prozess-Steuerung

BMBF Verbundprojekt

Förderkennzeichen: 01 IN 505 B

Laufzeit: 1. Oktober 1995 bis 31. Dezember 1999

Abschlussbericht

Prof. Dr.–Ing. Peter Protzel

Dr. rer. pol. Achim Lewandowski

Dipl.–Phys. Lars Kindermann

Dipl.–Inform. Michael Tagscherer

Dr.-Ing. Bärbel Herrnberger

Projektpartner:

Bayerisches Forschungszentrum für Wissensbasierte Systeme (FORWISS)

Berliner Wasserbetriebe (BWB)

Institut für Neuroinformatik der Universität Bochum (INI)

Siemens AG (Koordinator) (SAG)

Zentrum für Neuroinformatik GmbH (ZN)

Inhaltsverzeichnis

1 Ziele und Ablauf des Vorhabens

Das Projekt AENEAS - Anwendung und Entwicklung Neuronaler Verfahren zur Autonomen Prozess-Steuerung - wurde vom 1.10.1995 bis zum 31.12.1999 vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) als Verbundprojekt im Rahmen des Förderschwerpunkts „Intelligente Systeme“ gefördert. Im Mittelpunkt des Vorhabens stand das Bemühen, moderne Verfahren der Informatik für industriell relevante, komplexe Prozesse nutzbar zu machen. Insbesondere sollten die Methoden der Neuroinformatik in Kombination mit Verfahren der künstlichen Intelligenz, der Mess- und Regelungstechnik und der statistischen Signaltheorie zu wesentlichen Ressourceneinsparungen in diesen Prozessen führen. Die Forschungsziele des Vorhabens wurden durch die Bedürfnisse der Anwendung geleitet und die Ergebnisse sollten anhand von zwei wichtigen Prozesstypen demonstriert werden, der Stahlverarbeitung und der Wasserwirtschaft. Beide Prozesse haben eine hohe volkswirtschaftliche Bedeutung, so steht z.B. die deutsche Stahlindustrie unter einem enormen internationalen Wettbewerbsdruck, der eine immer kostengünstigere und gleichzeitig qualitativ hochwertigere Produktion erfordert.

Partner in dem Verbundprojekt waren neben dem Bayerischen Forschungszentrum für Wissensbasierte Systeme (FORWISS) die Siemens AG als Projektkoordinator, das Institut für Neuroinformatik der Ruhr-Universität Bochum, das Zentrum für Neuroinformatik GmbH in Bochum sowie die Berliner Wasserbetriebe. In dem vorliegenden Bericht werden die Arbeiten beschrieben, die das FORWISS im Rahmen von AENEAS durchgeführt hat.

1.1 Problemstellung und wissenschaftliche Herausforderungen

Die Arbeiten am FORWISS wurden in enger Kooperation mit der Siemens AG (Bereiche Zentrale Forschung und Anlagentechnik) durchgeführt und konzentrierten sich auf das Anwendungsfeld der Stahlverarbeitung. Die dort ablaufenden und zu optimierenden Prozesse sind charakterisiert durch starke Nichtlinearitäten, die sich einer analytischen Beschreibung weitgehend entziehen, sowie durch ein zeitvariantes Verhalten, bei dem sich die Nichtlinearitäten auch kurzzeitig stark ändern können. Voraussetzung für die Optimierung der Prozessabläufe ist jedoch immer eine ausreichend genaue Modellbildung, die dann z.B. Prognosen für vorausschauende Steuereingriffe liefern kann.

Neuronale Netze sind prinzipiell in der Lage, solche nichtlinearen Prozesse zu modellieren, falls ausreichend Trainingsdaten zur Verfügung stehen. Diese Voraussetzung war im Bereich der Stahlverarbeitung aufgrund des großen Materialdurchsatzes glücklicherweise erfüllt. Die wissenschaftliche Herausforderung besteht jedoch darin, dass die Neuronalen Netze aufgrund der Zeitvarianz der Prozesse nicht nur einmalig (off-line) trainiert, sondern im laufenden Betrieb (on-line) ständig „nachtrainiert“ werden müssen. Diese „on-line Adaptionsfähigkeit“, die wir als *kontinuierliches Lernen* bezeichen, war jedoch keineswegs Stand der Technik. Typisch für den industriellen Einsatz Neuronaler Netze war und ist das einmalige Trainieren vor dem eigentlichen Einsatz. Sobald ein Neuronales Netz nachtrainiert wird, führt dies auf das Stabilitäts- vs. Plastizitätsdilemma, das im zweiten Abschnitt illustriert wird. Ein Großteil der Forschungsarbeiten konzentrierte sich daher auf die Entwicklung neuer Verfahren des kontinuierlichen Lernens.

Eine zweite wissenschaftliche Herausforderung wurde ebenfalls durch die Anwendung motiviert und bestand in der Modellierung von Größen, die nur indirekt messbar sind. Konkret bestand eine der Anwendungsaufgaben darin, Profilwerte des Stahlblechs in einer Walzstraße nach jedem Walzgerüst zu modellieren, wobei ein Messwert jedoch nur nach dem letzten Gerüst zur Verfügung stand. Dieses Problem führte in der Theorie auf die Bestimmung von Funktionswurzeln. Mit dem Einsatz von Neuronalen Netzen zur Ermittlung von Funktionswurzeln in nichtlinearen Systemen wurde völliges Neuland betreten. Da solche Fragestellungen bei allen iterierten Prozessen auftreten können, geht der potentielle Anwendungsbereich der entwickelten Verfahren weit über die im Projekt beispielhaft verfolgten industriellen Anwendungen hinaus. Das gleiche gilt für die entwickelten Verfahren des kontinuierlichen Lernens.

Als Orientierungshilfe wird im folgenden ein kurzer Überblick über die durchgeführten Arbeiten und die Inhalte des Berichts gegeben.

1.2 Überblick über die durchgeführten Arbeiten

Der zweite Abschnitt des vorliegenden Berichts behandelt das Thema des kontinuierlichen Lernens, wobei zunächst genau definiert wird, was wir darunter verstehen und worin das Problem besteht. Das damit eng verbundene und bekannte Stabilitäts- vs. Plastizitätsdilemma wird anschließend erläutert und anhand eines Beispiels grafisch veranschaulicht. Der Stand der Technik beim kontinuierlichen Lernen wird in Abschnitt 2.3 zusammengefasst, wobei die gängigen Verfahren (gleitende Fenster, periodisches Netztraining bzw. Aufteilung des Arbeitsraums) kurz erläutert werden. Anschließend werden in Abschnitt 2.4 eine Reihe neuer Ansätze vorgestellt, die im Rahmen des Projekts erarbeitet wurden. Hier wurden insbesondere neue Verfahren zur adaptiven Raumaufteilung mit flexiblen Metriken entwickelt. Die praktische Anwendung dieser Verfahren und die dabei erzielten Ergebnisse werden im Abschnitt 5 dargestellt.

Ein zentrales Ergebnis des Projekts ist die Entwicklung einer neuartigen Neuronalen Netzarchitektur, die für das kontinuierliche Lernen besonders geeignet ist. Die Idee des Verfahrens basiert auf einer zweischichtigen Architektur mit einer Raumaufteilungsschicht und einer Schicht mit lokalen linearen oder nichtlinearen Modellen. Im Gegensatz zu bekannten Verfahren mit ähnlicher Architektur besteht hier eine enge Interaktion zwischen den beiden Schichten, so dass z.B. die Prognosefehler der lokalen Modelle eine Rückwirkung auf die Raumaufteilung haben. Die Struktur des Netzes wird von der Präsentation des ersten Datensatzes an inkrementell aufgebaut und das Netz liefert sofort nach Beginn des Trainings verwertbare Ergebnisse (Initiallernen). Sowohl die lokalen Modelle als auch die Raumaufteilung ist während der gesamten Einsatzzeit adaptiv, was auch das mögliche Schrumpfen des Netzes miteinbezieht. Ein weiterer wesentlicher Vorteil gegenüber anderen Lernverfahren ist die geringe Zahl einzustellender Parameter; so müssen z.B. keine Topologieparameter eingestellt werden. Neben der eigentlichen Ausgabe des Netzes erlaubt das Verfahrens auch eine Abschätzung der Vertrauenswürdigkeit für jeden Ausgabewert. Der Lernalgorithmus mit der Bezeichnung „Incremental Clustering and Evolving/Evaluation - ICE“ wird in Abschnitt 3.2. dargestellt und das Lernverhalten wird anschließend anhand von Beispielen verdeutlicht.

Als weiteres zentrales Ergebnis folgen im vierten Abschnitt die theoretischen Betrachtungen zur Analyse iterierter Prozesse. Diese Arbeiten wurden wie bereits geschildert durch die praktische Aufgabenstellung motiviert, haben aber darüberhinaus vielfältige Anwendungsmöglich-

keiten. In Abschnitt 4.1 werden zunächst die Begriffe erläutert und die Existenz und Eindeutigkeit möglicher Lösungen diskutiert. Für den einfacheren Fall linearer Funktionen werden in Abschnitt 4.2 analytische Lösungen betrachtet. Völlig neu ist der Einsatz Neuronaler Netze zur näherungsweisen Berechnung von Funktionswurzeln beliebiger nichtlinearer Funktionen, die lediglich durch Abtastwerte (Messwerte) beschrieben sind. In Abschnitt 4.3 sind verschiedene Ansätze hierzu mit ihren Vor- und Nachteilen beschrieben. Anschließend werden mögliche Anwendungsfelder aufgezeigt, die von der Modellierung iterativer technischer Prozesse über geometrische Abbildungen bis zur Analyse chaotischer Systeme reichen. Der Einsatz dieser Verfahren zur Modellierung von Zwischenprofilen in einer Walzstraße wird ausführlich im nächsten Abschnitt dargestellt.

Nach der Darstellung der grundlagenorientierten Arbeiten beschreibt der fünfte Abschnitt die Ergebnisse einer beispielhaften industriellen Anwendung im Bereich der Walzwerksautomatisierung. In Zusammenarbeit mit dem Siemens Unternehmensbereich Anlagentechnik wurde die Aufgabe der Modellierung und Prognose von Profilen warmgewalzter Stahlbänder bearbeitet. Diese Aufgabenstellung vereinigt alle Probleme, die in den grundlagenorientierten Arbeiten behandelt wurden. Neben der Nichtlinearität und Zeitvarianz der Prozesse besteht hier das Problem, das ein Profil nur am Ende der Walzstraße gemessen werden kann, aber auch sogenannte Zwischenprofilwerte hinter jedem Gerüst benötigt werden, um die Anlagensteuerung zu optimieren. In Abschnitt 5.2 werden zunächst Verfahren zur Modellierung und Prognose des Endprofils mit ihren Ergebnissen vorgestellt. Abschnitt 5.3 wendet dann die theoretischen Ergebnisse der Funktionswurzelbestimmung auf das Problem der Zwischenprofilprognose an und diskutiert die Grenzen des Verfahrens. Eine anwendungsspezifische elegante Lösung wurde schließlich in einem Hybridsystem realisiert, bei dem ein Neuronales Netz das Endprofil und ein lineares Modell die Zwischenprofile modelliert. Beide Verfahren adaptieren dabei kontinuierlich und die Ergebnisse des Endprofilnetzes werden zur Optimierung der Zwischenprofilprognose verwendet.

Der Bericht schließt in den Abschnitten 6 und 7 mit einer Zusammenfassung und der Angabe der verwendeten Literatur sowie der im Laufe des Projekts entstandenen Veröffentlichungen.

2 Methoden des kontinuierlichen Lernens

2.1 Definition von Aufgabe und Begriffen

Neuronale Netze werden in der Prozesssteuerung und -regelung typischerweise dann eingesetzt, wenn kein oder kein ausreichend genaues Streckenmodell existiert, z.B. weil die Nichtlinearitäten des Prozesses nicht genau bekannt und daher nicht analytisch fassbar sind. Eine der Grundaufgaben Neuronaler Netze in der Prozessautomatisierung ist daher die Identifikation von nichtlinearen Prozessen. Als Resultat erhält man in der Regel die Modellierung eines Kennfeldes, das z.B. eine Ausgangsgröße y des Prozesses als Funktion des Eingangsvektors **x** beschreibt. Fragen der Dynamik von Prozessen, d.h. die zeitliche Abhängigkeit des Ausgangs nicht nur von aktuellen, sondern auch von vergangenen Eingangswerten, werden im folgenden ausgeklammert, da sie eine zusätzliche Problem-Dimension darstellen. Die Aufgabe der Identifikation nichtlinearer Prozesse wird erheblich erschwert, wenn der Prozess zeitvariant ist, d.h. wenn sein Kennfeld auch von der Zeit t abhängt [1]. Um das Prinzip zu verdeutlichen, betrachten wir im folgenden nur eine Eingangsgröße x. Bezüglich der Ausgangsgröße eines Prozesses $y(x,t)$ überlagern sich dann mehrere Effekte:

a)Nichtlinearität des Kennfeldes $y(x) \neq ax + b$,

b)Zeitvariantes Prozessverhalten $y(x,t_1) \neq y(x,t_2)$,

c)Arbeitspunktverschiebungen der Eingangsgrößen $x = x(t)$.

Punkt a) entspricht der Approximation einer nichtlinearen, zeitinvarianten Funktion und stellt die „klassische“ Anwendung für Neuronale Netze dar. Die Netze können das nichtlineare Kennfeld $y(x)$ im Prinzip beliebig genau nachbilden, falls ausreichend Trainingsdaten zur Verfügung stehen, was wiederum durch eine entsprechende Arbeitspunktvariation $x(t)$ (Punkt c) gewährleistet werden muss. Typischerweise wird das Netz in diesem Fall einmalig trainiert und im anschließenden Einsatz nicht mehr verändert. Ist der Prozess zusätzlich zeitvariant (Punkt b), ist eine schritthaltende Adaption des Neuronalen Netzes erforderlich. D.h., ein einmaliges Training vor dem „Einsatz“ des Neuronalen Netzes ist nicht mehr ausreichend, sondern es muss während des Einsatzes kontinuierlich „nachtrainiert“ werden. Im Gegensatz zum zeitlich begrenzten Training bezeichnen wir dies als zeitlich unbegrenztes oder *kontinuierliches Lernen*. Andere Bezeichnungen wie on-line Lernen, Adaption, sequentielles oder inkrementelles Lernen sind ebenfalls geläufig, aber nicht eindeutig definiert und können je nach Kontext zu Fehldeutungen führen.

Das Grundproblem des kontinuierlichen Lernens einer nichtlinearen, zeitvarianten Funktion soll anhand der Abbildungen veranschaulicht werden. Abb. 1a) zeigt zunächst einen angenommenen zeitlichen Verlauf der Eingangsgröße $x(t)$, die z.B. die Arbeitspunktverschiebung eines Prozesses beschreibt. In diesem Beispiel wird der gesamte Wertebereich von x im Laufe der Zeit mehrfach „durchfahren“. Abb. 1b) zeigt als Beispiel die zeitliche Veränderung eines nichtlinearen Zusammenhangs zwischen Eingangs- und Ausgangsgröße eines Prozesses. Dies ist dargestellt anhand von zwei „Schnappschüssen“ des Kennfeldes zum Zeitpunkt t_0 und zum Zeitpunkt t_0+t. In der Praxis sind dieses Kennfelder nicht bekannt, sondern es liegen lediglich

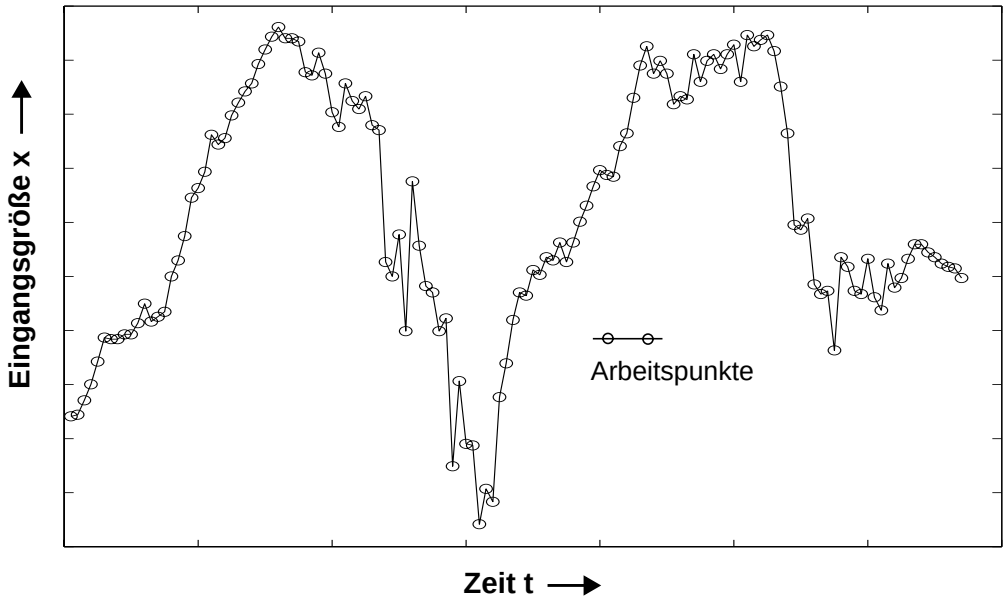


ABBILDUNG 1: a) Arbeitspunktverlauf $x(t)$ eines beispielhaften Prozesses.

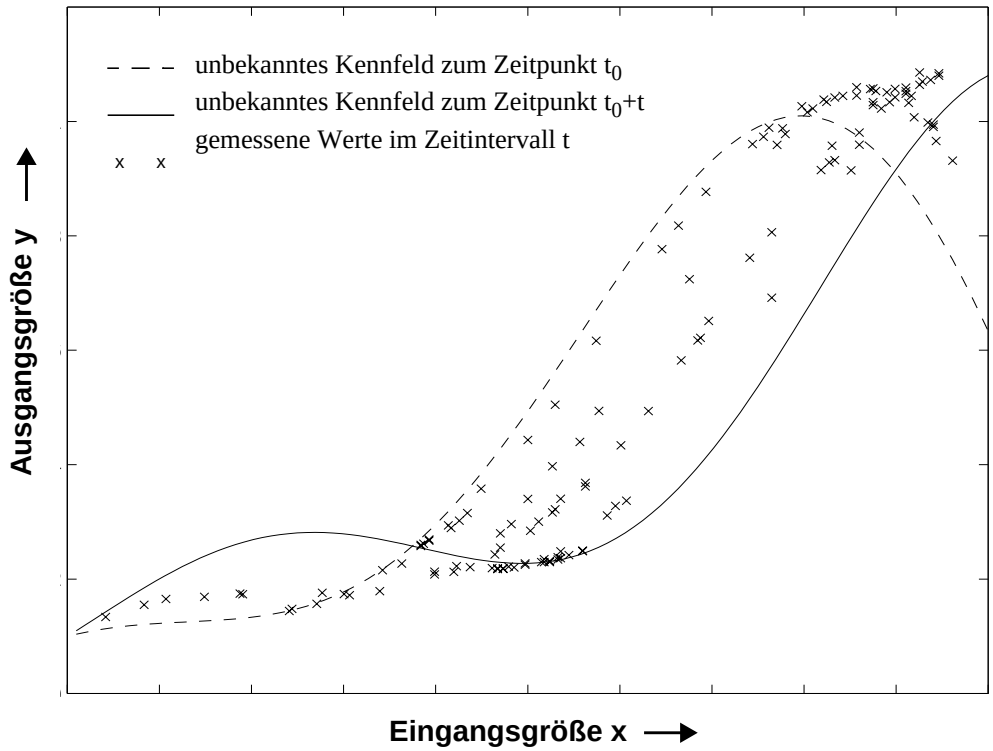


ABBILDUNG 1: b) Darstellung der Daten, die sich aus der zeitlichen Veränderung eines nichtlinearen Kennfelds und aus dem Arbeitspunktverlauf nach 1a) ergeben. Ohne Berücksichtigung der Zeitstruktur scheinen die Daten verrauscht zu sein.

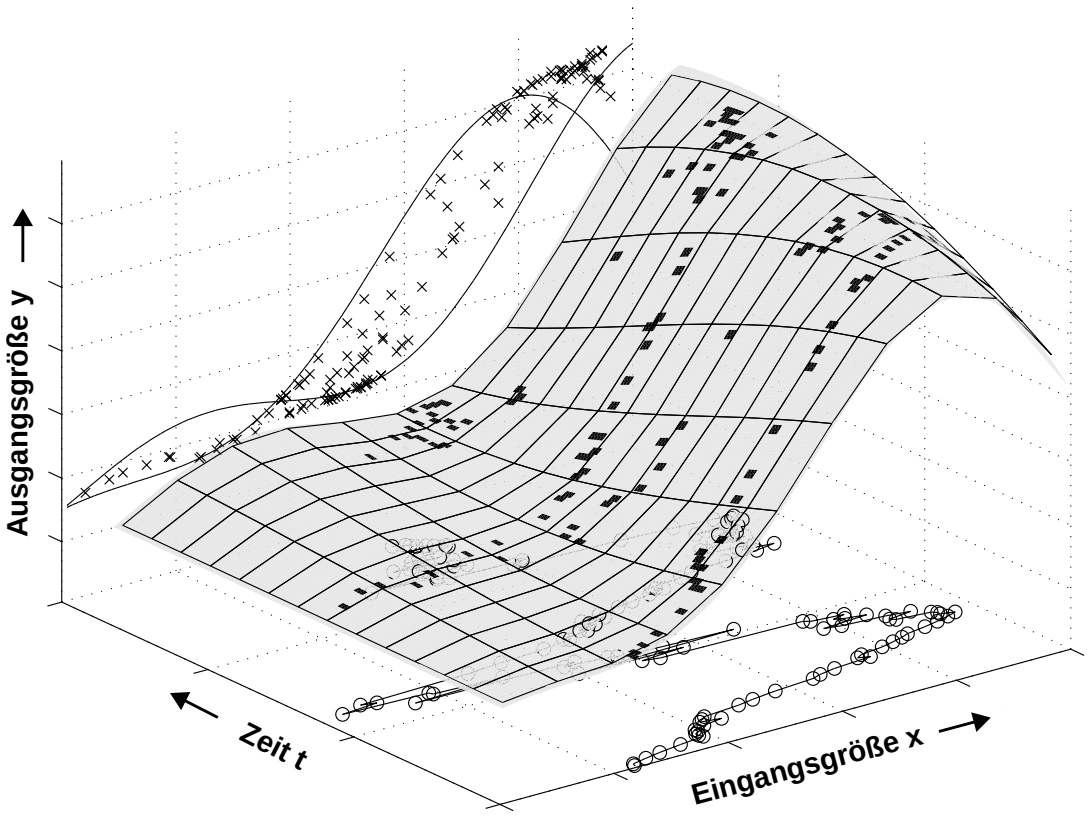


ABBILDUNG 2: Beispiel einer nichtlinearen, zeitvarianten Funktion und zur Modellierung dieser Funktion verfügbarer Daten (dunkle Flächen) entsprechend eines gegebenen Arbeitspunktverlaufs. Nur aus den bis zu einem Zeitpunkt t gesehenen Daten soll ein Modell des Kennfelds zum Zeitpunkt t+1 „erlernt“ werden. Die zeitliche Struktur der Daten kann und muß dabei berücksichtigt werden.

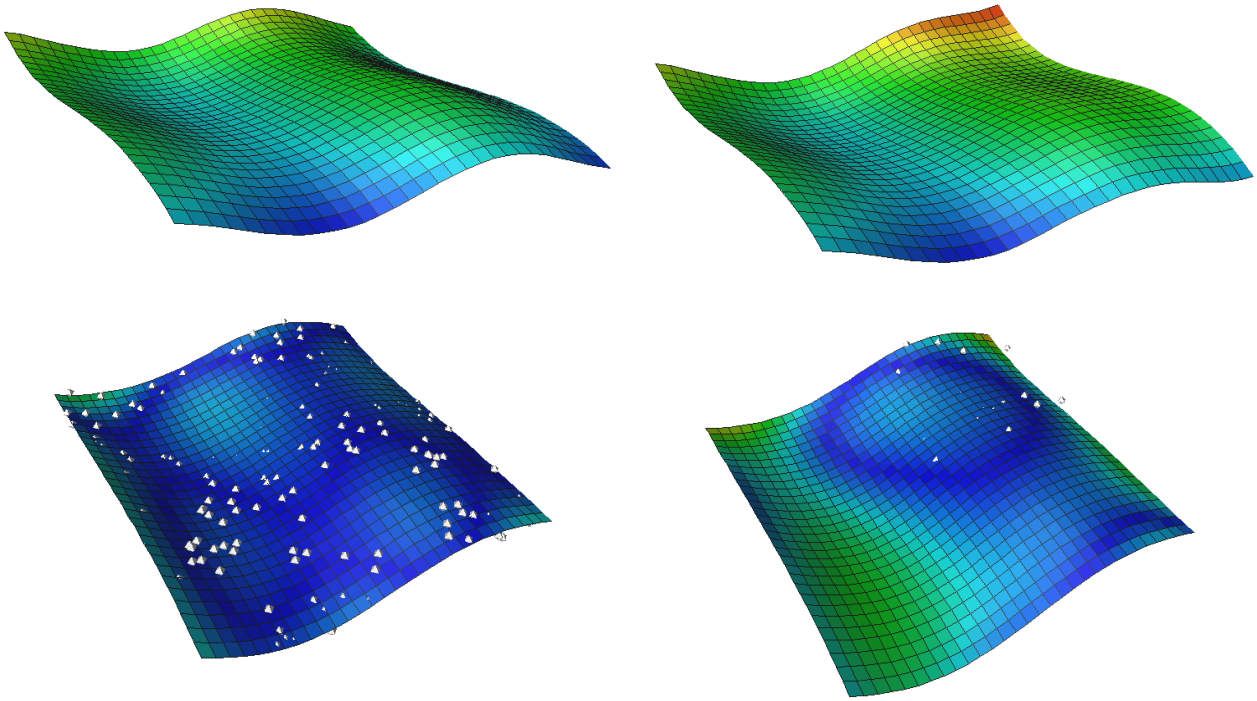
Messwerte vor, die zusätzlich mit Rauschen behaftet sein können. In Abb. 1b) sind die Messwerte eingetragen, die sich entsprechend des zeitvarianten Kennfeldes und des Arbeitspunktverlaufs nach Abb. 1a) ergeben.

Die dreidimensionale Darstellung des zeitvarianten Kennfeldes $y(x,t)$ in Abb. 2 verdeutlicht das Gesamtproblem: Die zur Modellierung verfügbaren Daten sind als dunkle Flächen auf dem (unbekannten) Kennfeld markiert. Abb. 1 a) und b) finden sich entsprechend als Projektionen auf die Grund- bzw. Seitenfläche wieder. Nur aus den bis zu einem Zeitpunkt t gesehenen Daten soll ein Modell des Kennfelds zum Zeitpunkt t+1 „erlernt“ werden. Die zeitliche Struktur der Daten kann und muss dabei berücksichtigt werden. Wenn alle Daten „gleichberechtigt“, d.h. ohne Berücksichtigung der Zeitstruktur, in die Modellierung eingehen, entsteht ein „Durchschnittsmodell“, das über die zeitliche Veränderung des Kennfeldes mittelt. Dieses Modell entspräche dann einem gemitteltem Kennfeld aus allen Datenpunkten in Abb. 1b), wobei die unverrauschten Daten durch das Ignorieren der Zeitstruktur verrauscht erscheinen.

2.2 Stabilitäts- vs. Plastizitätsdilemma

Beim kontinuierlichen Lernen treten eine Reihe von bekannten Problemen auf, z.B. wie kann sichergestellt werden, dass das Neuronale Netz relevante neue Daten möglichst schnell lernt, ohne alte, immer noch relevante Informationen zu vergessen? (Stabilitäts- vs. Plastizitätsdilemma, [2]). Dies ist insbesondere dann problematisch, wenn die auf Seite 9 genannten Effekte b) und c) auf ähnlichen Zeitskalen auftreten. Leider kommt dies in industriellen Anwendungen nicht selten vor; man spricht auch von der „Tagesform“ einer Anlage, die ein Bediener manchmal erkennen kann, der ein Prozessmodell jedoch automatisch folgen muss. Neuronale Netze müssen in der Lage sein, neues Wissen zu erlernen bzw. sich an Veränderungen anzupassen und gleichzeitig bereits akquiriertes Wissen nicht zu "verlernen".

Eine adaptive Anpassung an den letzten gesehenen Daten kann zur Folge haben, dass eine bereits perfekte Anpassung in anderen Teilen des Arbeitsraumes verloren geht. Diese Problematik wird als Catastrophic Interference bezeichnet. Der Effekt wird beeinflusst durch die Art und Weise, wie die Daten in Neuronalen Netzen repräsentiert werden, und durch die speziellen Eigenschaften einzelner Netzkomponenten. In Abb. 3 ist der Effekt der Catastrophic Interference am Beispiel eines MLPs mit sigmoiden Aktivierungsfunktionen visualisiert. Links ist die zu-



(a) vor der Prozessdrift (b) während der Prozessdrift

ABBILDUNG 3: Catastrophic Interference – am Beispiel eines MLPs. Oben ist jeweils die Zielfunktion vor bzw. während der Prozessdrift dargestellt. Darunter sind die entsprechenden abgeleiteten Modelle visualisiert. Während die Zielfunktionen nach der Höhe eingefärbt sind, entspricht die Helligkeit der Modellfläche dem lokalen Fehler (dunkel-kleiner Fehler / hell - großer Fehler).

nächst zeitinvariante Zielfunktion und das vom MLP identifizierte Modell dargestellt. Für das Training werden 150 Beobachtungen gleichverteilt aus dem gesamten Eingaberaum selektiert.

Dies spiegelt sich in einem entsprechend kleinen Fehler im gesamten Bereich des Eingaberaums wider. Anschließend werden die rechten beiden Quadranten der Zielfunktion kontinuierlich über die Zeit angehoben (vgl. Abb. 3 oben). Während der Drift werden dem Netz lediglich Daten aus dem rechten hinteren Eingaberaumquadranten präsentiert. Deutlich ist in Abb. 3 (unten) zu erkennen, dass in diesem Bereich der kleinste Fehler beobachtet werden kann (dunkle Regionen).

Darüber hinaus zeigt sich jedoch, dass auch andere Bereiche während des Lernens modifiziert werden. Beispielsweise ist eine deutliche Veränderung im linken vorderen Quadranten zu beobachten. Der Fehler steigt in diesem Quadranten merklich an (helle Gebiete). Zu beachten ist, dass in diesem Bereich die Zielfunktion nicht verändert wird. Das heißt, bereits akquiriertes Wissen geht hier verloren - das Netz vergisst.

2.3 Bekannte Ansätze zum kontinuierlichen Lernen

Gleitendes Fenster

Ein naheliegende Methode besteht darin, ein Fenster auf den Daten zu definieren und dieses mit einer festen Schrittweite über die Daten zu schieben. Für das Netztraining werden jeweils die Daten innerhalb des gleitenden Fensters verwendet. Abbruchkriterien für das Training können beispielsweise eine feste Epochenzahl, eine vorgegebene Fehlerschwelle oder ein Signal sein, welches anzeigt, dass das Modell nun angewendet werden muss, um z.B. eine neue Prognose zu liefern.

Die Wahl der Parameter (Fenstergröße, Schrittweite, Abbruchkriterium usw.) hat einen entscheidenden Einfluss auf die Modellierungsgüte und die Stabilität des Modells und kann in der Regel nur durch Experimente an das jeweilige Problem angepasst werden. Wird beispielsweise das Modell durch eine zu kleine Fenstergröße und Fehlerschranke zu intensiv mit einem nicht repräsentativen Datensatz nachtrainiert, kann bereits vorhandenes „Wissen“ durch eine zu starke Adaption völlig verloren gehen. Insbesondere in dem Grenzfall, bei dem nur mit dem letzten, aktuellen Muster nachtrainiert wird, besteht die Gefahr, dass das Netz die eigentliche Funktion nicht identifiziert, sondern lediglich eine Offset-Anpassung vornimmt, um den Fehler jeweils auf dem einen Datenpunkt zu minimieren.

Periodisches Netztraining

Im Gegensatz zum gleitenden Fenster werden beim periodischen Netztraining die anfallenden Trainingsdaten über einen längeren Zeitraum gesammelt, um anschließend mit diesem „großen“ Datensatz das Netz nachzutrainieren. Feste Zeitschranken, das Erreichen einer vorgegebenen Datenmenge, Maschinenpausen oder Teilewechsel können Signale bzw. Situationen sein, durch die ein Netztraining gestartet wird.

Die größere Datenmenge soll sicherstellen, dass das Training mit repräsentativen Daten erfolgt und somit zu einer hohen Modellgüte führt. Wird jedoch das Intervall zwischen zwei Trainingseinheiten zu groß gewählt, können sich auch hier Probleme ergeben, wenn der Prozess zeitlich stark schwankt. Der Fehler, den das Netz als Prozessmodell bei der Anwendung liefert, steigt infolge der Zeitvarianz des Prozesses bis zum nächsten Nachtraining an, da das Netz erst dann wieder an die Veränderungen angepasst wird. Werden anschließend alle gesammelten Daten

gleichberechtigt, d.h. ohne Berücksichtigung der Zeitstruktur, zum Nachtraining verwendet, wird ein über die zeitliche Veränderung des Prozesses gemitteltes Modell gelernt, wie bereits im ersten Abschnitt (Abb. 1 u. Abb. 2) erläutert wurde.

Um eine höhere Plastizität zwischen zwei Trainingsabschnitten zu erreichen, kann periodisches Netztraining z.B. mit einem gleitenden Fensteransatz kombiniert werden. Durch die permanente Präsentation aktueller Daten (gleitendes Fenster) erfährt des Netz eine Anpassung an den aktuellen Zustand des Prozesses. Die oben beschriebenen Probleme der Parametereinstellung und des periodischen Nachtrainings bleiben jedoch weitgehend erhalten.

Aufteilung des Arbeitsraums

Bei zeitvarianten Prozessen kommt es häufig vor, dass sich die Prozesscharakteristik nur lokal, d.h. in bestimmten Bereichen des Raums der Eingangsgrößen (Arbeitsraum) ändert. Für die Stabilität beim kontinuierlichen Lernen ist es daher günstig, wenn mehrere Modelle für die verschiedenen Arbeitsraumbereiche existieren und bei lokalen Veränderungen dann nur das jeweils „zuständige“ Modell angepasst werden muss. Neuronale Netze kommen dieser Anforderung nur bedingt entgegen; es gibt sowohl Netze mit globalem Verhalten (z.B. Multilayer-Perceptrons - MLPs) als auch Netze mit lokalen Eigenschaften (z.B. Netze mit radialen Basisfunktionen - RBF Netze).

Multilayer-Perzeptrons (MLPs) mit sigmoider Aktivierungsfunktion zeigen globales Verhalten in dem Sinne, dass sich die Aktivierung eines Neurons und damit auch die Veränderung von Gewichten im gesamten Raum der Eingangsgrößen (Arbeitsraum) auswirkt. Beim kontinuierlichen Lernen hat das zur Folge, dass eine (erwünschte) Anpassung des Netzes an eine lokale Prozessänderung auch (unerwünschte) Auswirkungen auf die gelernte Funktion in allen anderen Raumbereichen nach sich ziehen kann. Die Aktivität der Neuronen von RBF-Netzen ist im Gegensatz dazu nur in einem definierten radialsymmetrischen Bereich um ihr Zentrum signifikant größer als Null. Daher wirken sich lokale Anpassungen beim Nachtraining auch nur lokal aus. Das Problem bei RBF-Netzen ist die große Zahl erforderlicher Neuronen, die mit der Dimension des Arbeitsraums exponentiell ansteigt.

Eine Alternative sind Hybrid-Verfahren, bei denen unterschiedliche Ansätze mit ihren Vorteilen kombiniert werden. Es gibt zahlreiche Ansätze mit zwei Verfahren, bei denen die erste Stufe eine Aufteilung des Arbeitsraums durchführt, um anschließend in der zweiten Stufe in den einzelnen Arbeitsraumregionen „lokale Modelle“ zu plazieren. So wird beispielsweise bei Counterpropagation-Netzen über ein Kohonen-Netz eine Aufteilung des Eingaberaums gemäß der Dichteverteilung der Eingangsdaten durchgeführt und über einen linearen Assoziator werden einfache lineare Modelle für die entsprechenden Regionen angepasst. Dabei wird die Kohonen-Schicht unüberwacht und der Lineare Assoziator überwacht trainiert, d.h. das Training des Assoziators ist vom Training der Kohonen-Schicht abgekoppelt. Da der Assoziator warten muss, bis das Kohonen-Netz eine sinnvolle Raumaufteilung gefunden hat, kann sich diese schwache Bindung zwischen den beiden Ansätzen nachteilig auswirken. So werden z.B. Raumbereiche mit relativ wenigen Datenpunkten durch den Kohonen Algorithmus nur grob oder gar nicht aufgeteilt, jedoch kann gerade in solchen Bereichen eine feine Modellierung mit mehreren Modellen aufgrund starker Nichtlinearitäten erforderlich sein.

2.4 Neue Ansätze

2.4.1 Kurzzeitadaption durch Fehlerextrapolation

Bei den folgenden Betrachtungen gehen wir von dem Szenario aus, dass ein neuronales Modell im Rahmen einer Prozessführung zur Prognose des aktuellen Prozesszustands eingesetzt wird. Dazu wird ein Eingangsvektor (Stellgrößen, Parameter etc.) an das Netz übergeben und dieses liefert gemäß der gelernten funktionalen Abbildung den Wert für eine oder mehrere Ausgangsgrößen als Prognose des Prozesszustands. Diese Prognose wird zur Prozessführung verwendet und nach einer bestimmten Zeit stehen Messwerte der „tatsächlichen“ Ausgangsgrößen zur Verfügung. Diese Messwerte können anschließend mit der Netzprognose verglichen und der resultierende Fehler kann im Rahmen eines kontinuierlichen Lernvorgangs zur Anpassung des neuronalen Modells verwendet werden.

Zur Realisierung von unterschiedlichen Adaptionsgeschwindigkeiten gibt es eine einfache Möglichkeit, zwei Verfahren zu kombinieren. Ein neuronales Modell liefert eine „Primärprognose“ und adaptiert relativ langsam, um die Nichtlinearitäten des Prozesses ausreichend modellieren zu können. Um trotzdem schnellen Änderungen des Prozesses folgen zu können, liefert ein zweites Verfahren eine Kurzzeitprognose, die darauf basiert, dass aufeinanderfolgende Fehler bei der Prognose statistisch korreliert sind. Eine solche Kurzzeitprognose kann beispielsweise durch eine „Fehlerextrapolation“ mit Hilfe sogenannter ARMA-Modelle nach Box-Jenkins erfolgen ([3]). Über die Autokorrelationsfunktion und die partielle Autokorrelationsfunktion können geeignete Modelle aus den Daten geschätzt werden. Ein Autoregressionsmodell 2. Ordnung (AR(2)-Modell) nimmt beispielsweise an, dass der Fehler $e(t) = p(t) - y(t)$ zwischen Prognose und tatsächlichem Ausgangswert (Messwert) zum Zeitpunkt t durch

$$e(t) = \Phi_1 \cdot e(t-1) + \Phi_2 \cdot e(t-2) + f(t)$$

dargestellt werden kann, mit $f(t)$ als weiterem Störterm. Der aktuelle Fehler hängt von den letzten beiden Fehlern ab. Die Koeffizienten Φ_1 und Φ_2 können aus den Daten geschätzt werden. Für AR-Modelle lassen sich rekursive Gleichungen verwenden, wodurch die Berechnung der aktuellen Parameterschätzer sehr schnell wird.

Sind die Prognosen und (tatsächlichen) Ausgangswerte bis zum Zeitpunkt $t-1$ bekannt und liegt die Primärprognose $p(t)$ vor, so ergibt sich die endgültige Voraussage zu $\hat{y}(t) = p(t) - \hat{e}(t)$, mit $\hat{e}(t)$ als prognostiziertem Fehler. Die Verwendung der Fehlerextrapolation hat in allen bisher von uns betrachteten realen Datensätzen aus industriellen Prozessen die Prognosegüte gegenüber einzelnen neuronalen Modellen um etwa 10-15 Prozent gesteigert. Die Abb. 4 zeigt das typische Bild bei Anwendung der Fehlerextrapolation. Während das primäre, langfristig adaptierende Prognoseverfahren deutliche Abweichungen zur Ausgangsgröße aufweist, liegen die mittels Fehlerextrapolation korrigierten, kurzfristig angepassten Prognosen sehr nahe an den tatsächlichen Werten.

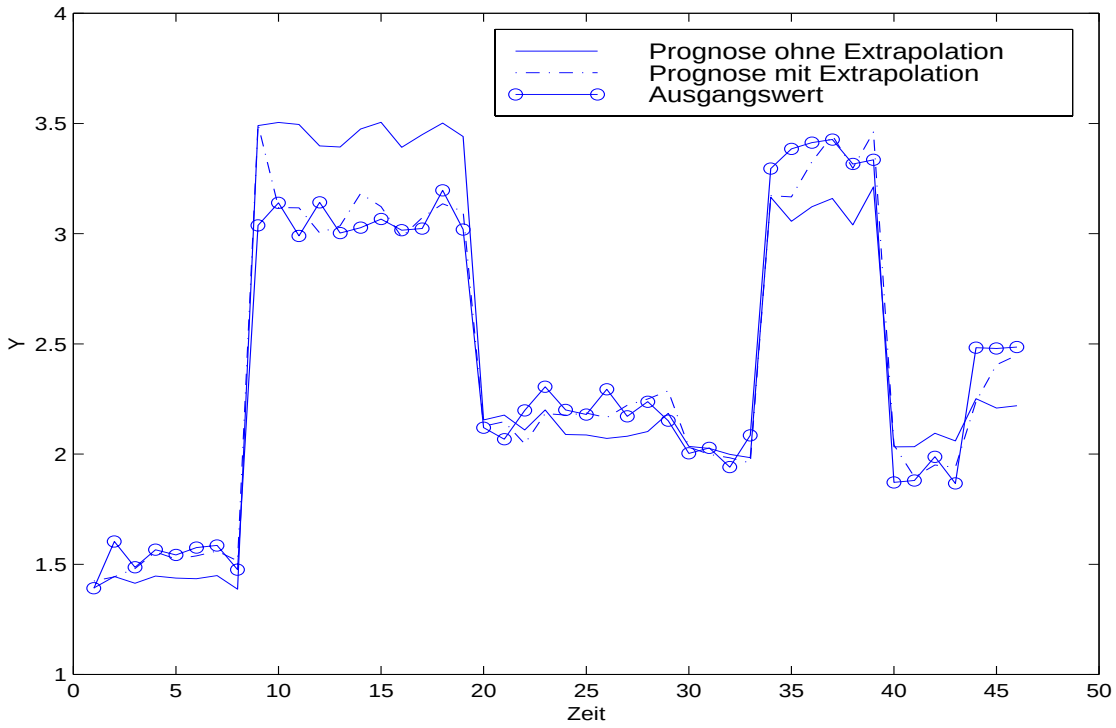


ABBILDUNG 4: Verbesserung der Prognosegüte durch Fehlerextrapolation mit einem AR(3)-Modell.

2.4.2 Adaptive Aufteilung des Arbeitsraums

Es sollen nun neue Ansätze betrachtet werden, die den Arbeitsraum aufteilen und diese Aufteilung auch daten- und fehlerabhängig im Laufe der Zeit ändern können. Die Grundidee der folgenden Beispiele ist es, Stützstellen über den Wertebereich der Eingangsvektoren zu verteilen und jeder Stützstelle ein lokales Modell zuzuordnen. Der Einzugsbereich jedes Modells wird über einen noch zu definierenden Abstand zum Eingangsvektor festgelegt; ein Modell ist für einen neu eintreffenden Datensatz dann zuständig, wenn seine Stützstelle die nächstgelegene ist. Als lokale Modelle werden lineare Modelle verwendet. Mit Hilfe dieser lokalen Modelle können bei genügend feiner Aufteilung des Arbeitsraumes auch nichtlineare Funktionen approximiert werden. Die vorgestellten Verfahren unterscheiden sich hauptsächlich in der Art und Weise, wie die Metrik zur Bestimmung des Abstands definiert wird. Die Adaption der Grenzen zwischen den einzelnen Modellen wird durch Verschiebung der Stützstellen und durch Adaption des Metrikmasses erreicht.

Standardisierte Euklidische Metrik

Der einfachste Fall wäre es, für den Abstand die Euklidische Metrik zu verwenden. Da der so definierte Abstand von den Einheiten der Variablen abhängt und eine einzige Variable den Abstand dominieren könnte, sollten die Variablen vorher standardisiert werden. Hierzu müssten natürlich Aussagen über die Streuung der Variablen vorliegen. Aber selbst eine standardisierte Norm beinhaltet den Nachteil, dass für die funktionale Abhängigkeit des Ausgangs unwichtige Eingangsvariablen als Rauschen in den Abstand eingehen. Ist z.B. der Ausgang von einer Variablen stark abhängig, von einer anderen jedoch nicht, so sollte das Modell besser sein, dessen

Stützstelle in der wichtigen Variablen dichter am Eingangsvektor ist. Diese Annahmen haben zur Entwicklung einer automatisch adaptierten Norm bzw. der daraus abgeleiteten Metrik geführt.

Automatisch generierte Metrik

Der Ablauf bei der Adaption des Abstandsmasses lässt sich folgendermaßen beschreiben. Ausgehend von einer Ausgangsabstandsmaß, dessen Quadrat durch $a = (x-s)^T D(x-s)$ dargestellt werden kann, mit der Diagonalmatrix D, x als Eingangsvektor und s als Stützstellenvektor, wird die Diagonalmatrix mit jedem neuen Datensatz verändert. Dazu wird bei Präsentation eines Eingangsvektors zunächst das Modell ermittelt, das gemäß der aktuellen Norm den geringsten Abstand besitzt. Mittels dieses Modells wird die Prognose erstellt und das Modell (also die lineare Funktion für diesen Bereich) adaptiert. Ist der Ausgang bekannt, wird nachträglich das Modell gesucht, das diesen Ausgang am besten vorausgesagt hätte. Enthält die beste Stützstelle eine Variable, die zur entsprechenden Variablen des Eingangssvektors den geringsten einfachen Abstand besitzt, im Vergleich zu den anderen Stützstellen, wird das zugehörige Diagonalelement mit einem Faktor grösser als eins multipliziert: $d_i^{neu} = (1+\varepsilon)d_i$ Gehört eine Variable zum zweitbesten Modell, bleibt das Diagonalelement unverändert. Die restlichen Diagonalelemente werden durch $(1+\varepsilon)$ dividiert. An dieser Stelle könnten natürlich auch andere Adaptionsvarianten ausgetestet werden.

Falls sich der Arbeitsraum im Laufe der Zeit verschiebt oder ausdehnt, kann es sinnvoll sein, die Stützstellen beweglich zu halten. In dieser Arbeit wird die beste Stützstelle immer ein wenig in Richtung Datensatz geschoben: $s_{neu} = s_{alt} + \lambda(x - s_{alt})$. Somit muss der Anwender bei diesem Ansatz die Anzahl und Initiallage der Stützstellen festlegen, sowie geeignete Werte der Parameter λ und ε finden.

Ein Vergleich mit der einfachen Euklidschen Metrik

Es sollen zunächst an einem einfachen Beispiel die unterschiedlichen Funktionsweisen der beiden genannten Metriken demonstriert werden. Die in der folgenden Abb. 5 dargestellte Funktion soll gelernt werden. Die Eingangsvektoren sind zufällig gleichverteilt, jede Variable auf dem Intervall von -2 bis +2. Eine Standardisierung muss also nicht vorgenommen werden. 10 Stützstellen wurden, ebenfalls zufällig, ausgestreut.

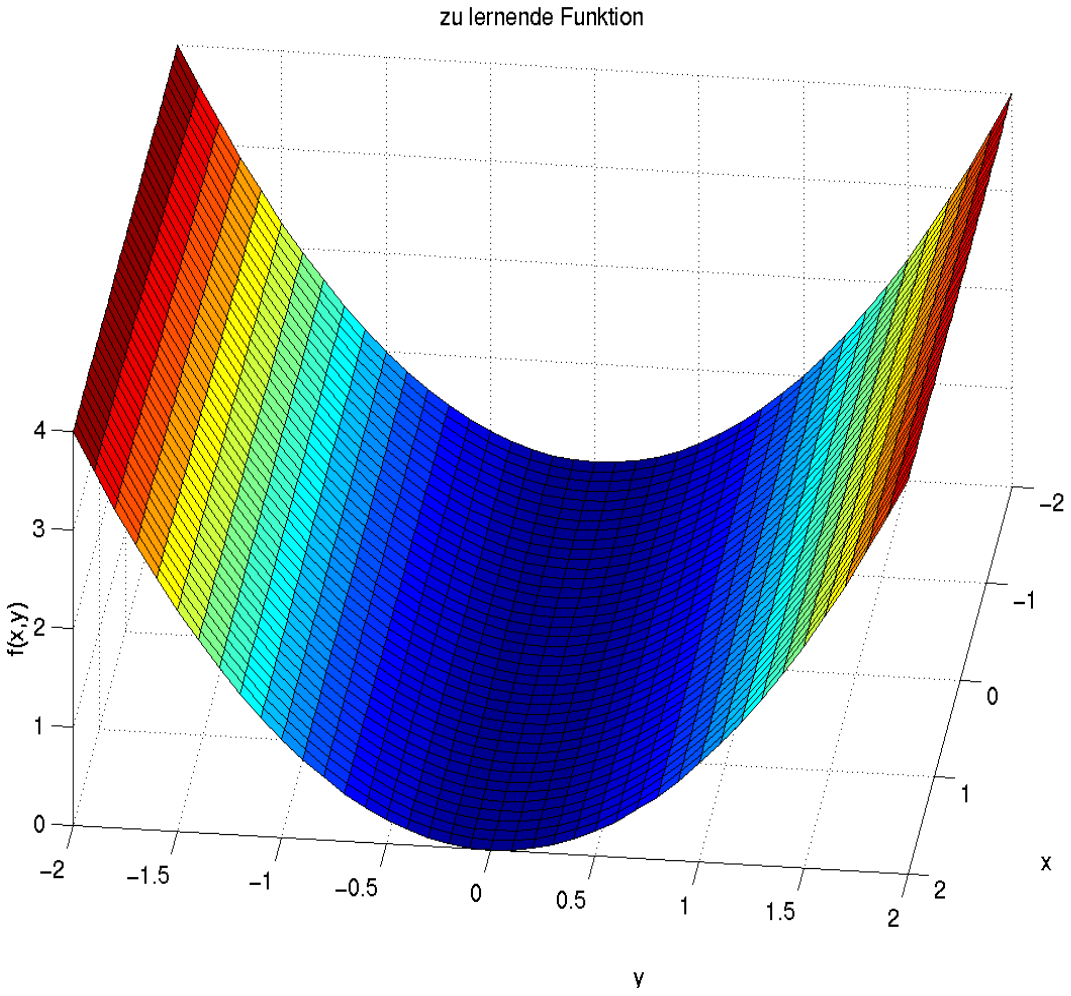


ABBILDUNG 5: Beispielfunktion zur Illustration

Die Abb. 6 und Abb. 7 zeigen jeweils links, wie sich der Eingangsraum auf die Modelle aufteilt (die Stützstellen sind in diesem Beispiel fix!), und rechts die Approximation der Funktion nach 2000 präsentierten Mustern. Deutlich sichtbar ist die Verbesserung der Approximation mittels der automatisch generierten Metrik (unteres Bild). Jedes Modell ist für einen Streifen in Richtung der x-Achse zuständig. Das Verfahren hat automatisch erkannt, dass es wichtiger ist, zu einem Eingangsvektor die nächstgelegene Stützstelle in y-Richtung zu finden.

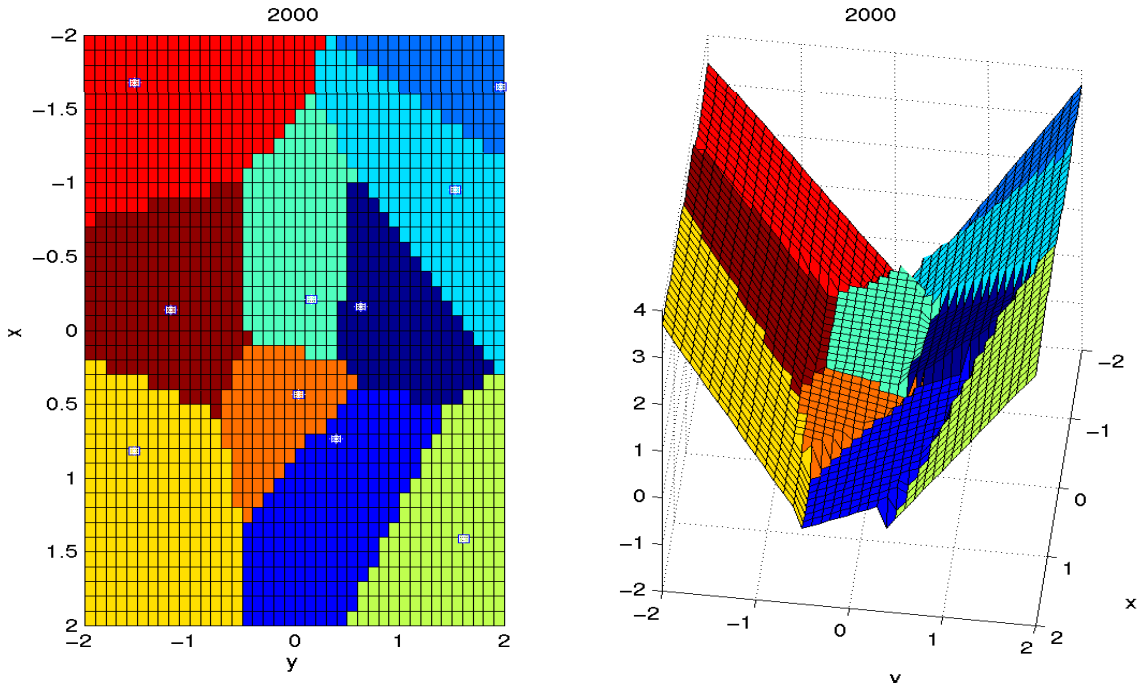


ABBILDUNG 6: Raumaufteilung mittels des euklidischen Abstands

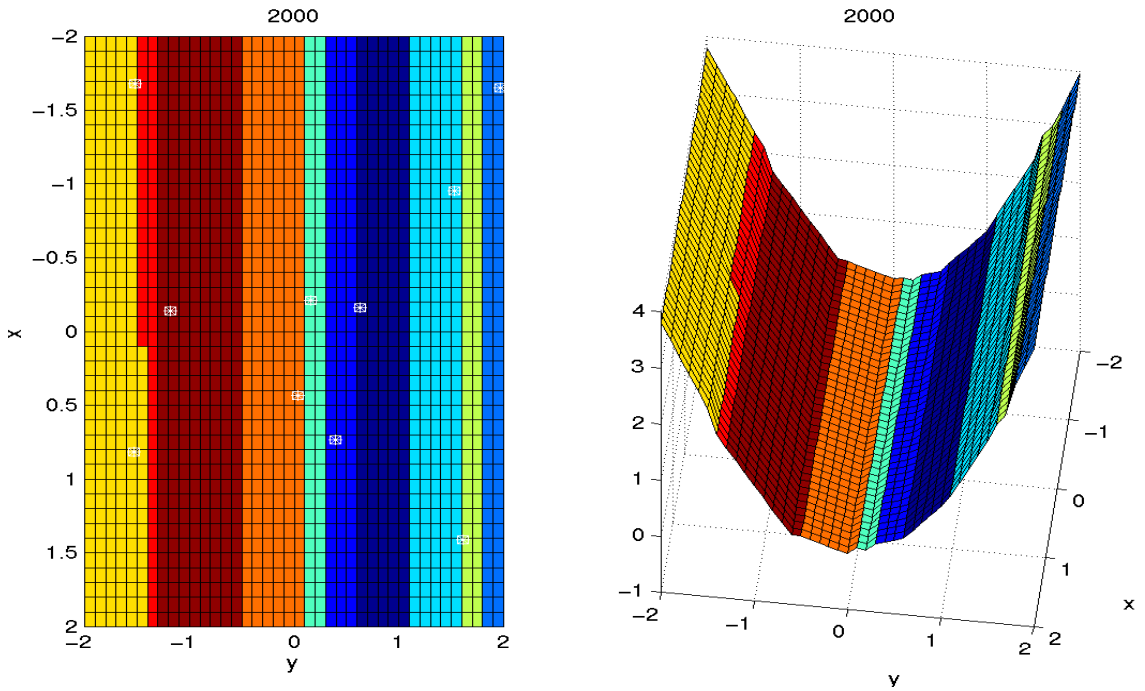


ABBILDUNG 7: Raumaufteilung mittels Autonorm

Anstelle der linearen Modelle pro Stützstelle können auch andere Modelle verwendet werden. Relativ einfach zu handhaben sind beispeilsweise quadratische Modelle, gemeint sind hier Modelle, die zusätzlich zu den eigentlichen Variablen noch deren Quadrate oder Kubiken verwenden.

Raumaufteilung mit flexiblen Metriken

Da bei dem automatisch generierten Abstandsmaß letztendlich nur ein globales Maß verwendet wird, werden möglicherweise lokale Besonderheiten der funktionalen Abhängigkeiten nicht angemessen berücksichtigt. Um diesen potentiellen Nachteil auszumerzen, wurde eine weitere Variante entwickelt, bei der jedes Modell eine eigene Metrik besitzt. Anschaulich gesagt soll sich ein Modell von seiner Stützstelle aus in die Regionen ausbreiten, in denen gute Prognosen erstellt wurden und sich aus den Regionen zurückziehen, wo die Voraussagen nicht befriedigend waren. Dies wird wieder über die Adaption der persönlichen Metriken erreicht. Da die Anzahl der veränderbaren Parameter im Vergleich zum Verfahren des letzten Abschnitts deutlich höher ist, können bessere Ergebnisse erzielt werden. Beispielsweise können die Trenngrenzen zwischen zwei lokalen Modellen nun auch nichtlinear sein. Als Vorteil hat sich auch die alternative Interpretation des Abstands des Eingangsvektors von der Stützstelle als Aktivierung einer Aktivierungsfunktion, verankert an der Stützstelle, erwiesen. Diese Verwendung einer Aktivierungsfunktion erlaubt eine einfachere Beschreibung des Verschmelzens benachbarter Modelle, um eine stetige Approximation zu erreichen, im Gegensatz zum bisherigen Verfahren, immer nur *ein* lokales Modell zur Prognose zu verwenden.

Eine frühe Version des nachfolgenden Algorithmus wurde auch auf eine Reihe von verschiedenen, künstlichen Benchmark-Problemen angewendet, die im Rahmen eines auf der NIPS 98 von unserer Gruppe organisierten Workshops präsentiert wurden. Das Verfahren konnte sich bereits damals im Vergleich zu den internationalen Mitstreitern bei der Anwendung auf einen Datensatz mit 10 Eingängen gut behaupten.

Der Algorithmus: Modelle und Modellupdate

Jedes lokale Modell wird wieder durch ein lineares Modell gegeben: $y_k = \beta^t x$

Durch die Verwendung von linearen Modellen kann der Vorteil des schnellen Updates durch rekursive Gleichungen genutzt werden. Ein Problem ist der Aufbau einer Menge von lokalen Modellen. Falls neue Daten erscheinen, stellt sich die Frage, ob und wo neue Modelle eingefügt werden sollten. Hier konzentrieren wir uns aber auf das Problem, eine feste, gegebene Anzahl von Modellen mit ihren Einzugsbereichen möglichst geschickt auf den Eingangsraum zu verteilen, um die bestmögliche globale Modellapproximation zu erzielen.

Für einen gegebenen Eingangsvektor x werden die individuellen Prognosen $\hat{y}_k$ der lokalen Modelle in einer Linearkombination:

$$\hat{y} = \frac{\sum_k w_k \hat{y}_k}{\sum_k w_k}$$

verschmolzen. Die Gewichte w_k sind durch die Aktivierungen von Gauss-Funktionen gegeben:

$$w_k = \exp(-\frac{1}{2}(x - c_k)^T D_k (x - c_k))$$

mit den Stützstellen c_k und den positiv definiten $(n \times n)$-Matrizen D_k, die die Form der Aktivierungsfunktion beschreiben. Hier erkennt man auch gut den Zusammenhang zwischen dem bisher verwendeten Abstand und der Aktivierung:

$$(x - c_k)^T D_k (x - c_k)$$

ist der Term, der bisher als (quadrierter) Abstand bezeichnet wurde.

Nachdem ein neuer Datensatz eingetroffen ist, werden alle lokalen Modellgleichungen angepasst. Es wird eine gewichtete Regression pro Modell verwendet, wobei auch das Gewicht für diesen Datensatz durch die Aktivierung der zugehörigen Gauss-Funktion bestimmt wird. Im Gegensatz zum Verfahren „Autonorm“ , bei dem jeder Datensatz bei der Modellanpassung das gleiche Gewicht bekam, besitzen hier alle Datensätze unterschiedliche Gewichte. Nach Ljung & Söderström [4] existiert eine praktische Rekursionsformel. Falls alle Eingangsvektoren zeilenweise in der Matrix X, alle Ausgangsgrößen in dem Vector Y und die Gewichte in einer Diagonalmatrix W zusammengefasst werden, kann zunächst der Parametervektor eines Modells durch

$$\hat{\beta} = (X^T W X)^{-1} X^T W Y$$

geschätzt werden. Mit der Schreibweise $P = (X^T W X)^{-1}$ kann der neue Parametervektor $\hat{\beta}_{t+1}$ nach Präsentation eines neuen Musters (x, y) in inkrementeller Weise als

$$\hat{\beta}_{t+1} = \hat{\beta}_t + e w P_{t+1} x$$

ausgedrückt werden, mit

$$P_{t+1} = \frac{1}{\lambda}\left(P_t - \frac{P_t x x^T P_t}{\frac{\lambda}{w} + x^T P_t x}\right)$$

und

$$e = y - \beta_t^T x$$

Gleichzeitig wurde ein Vergessensfaktor λ eingefügt, um den Einfluss älterer Beobachtungen zu eliminieren, da sich während des Lernens die rezeptiven Felder verändern.

Update der rezeptiven Felder

Der wichtigste Teil das Algorithmus ist durch die Adaption der rezeptiven Felder gegeben. Nachdem ein Trainingsmuster präsentiert wurde, wird das Modell k mit der höchsten Aktivierung etwas in Richtung Eingangsvektor x verschoben, in Abhängigkeit vom Vorhersagefehler $e_{k,t+1}$:

$$c_{k,t+1} = c_{k,t} + \eta min\left(1, \frac{|e_{k,t+1}|}{e_{max}}\right)(x - c_{k,t})$$

Der Parameter $\eta > 0$ wird nahe bei null gewählt, und der Einfluss von Ausreissern wird in der Hinsicht beschränkt, dass absolute Fehler über einem Threshold von e_{max} abgeschnitten werden. Mit dieser Konstruktion sollen nur kleine, aber fehlerabhängige Veränderungen zugelassen werden, um die Stützstellen besser zu verteilen.

Der Clou des Verfahrens liegt nun darin, dass nur benachbarte Modelle des Eingangsvektors in das Update eingehen.

Für kleine Werte von M, etwa $M = 3$ oder $M = 5$, werden die M Modelle mit der höchsten Aktivierung, also im alten Sinne die nahesten Modelle, nach dem absoluten Fehler ihrer Voraussagen $\hat{y}_k$ für das letzte Trainingsmuster sortiert. Die Ränge dieser Modelle werden linear in $R = -1$ für das beste und $R = 1$ für das schlechteste Modell transformiert. Falls das Modell mit der höchsten Aktivierung die beste Vorhersage geliefert hatte, bleiben die Aktivierungen unverändert. Im anderen Falle wird der folgende Schritt für jede der Distanzmatrizen, die zu den ausgewählten M Modellen gehören, ausgeführt, wobei der Parameter $\rho > 0$ die Größe der Änderung steuert:

$$D_{k,t+1} = D_{k,t} + \frac{((1+\rho)^R - 1)}{(x-c_k)^T D_{k,t}(x-c_k)} D_{k,t}(x-c_k)(x-c_k)^T D_{k,t}$$

Falls zufällig x und c_k identisch sind, verändert sich die zugehörige Matrix nicht. Der entscheidende Teil der Aktivierungsfunktion,

$$p_{k,t} := (x-c_k)^T D_{k,t}(x-c_k)$$

w_k ist nach dem Update durch

$$p_{k,t+1} = (1+\rho)^R p_{k,t}$$

gegeben.

Die Aktivierung eines besseren Modells ($R < 0$) steigt somit, während die Aktivierung der schlechten Modelle ($R > 0$) sinkt. Weiterhin bleibt die Aktivierung eines weiteren Vektors z mit

$$(z-c_k)^T D_{k,t}(x-c_k) = 0$$

unverändert:

$$(z-c_k)^T D_{k,t+1}(z-c_k) = (z-c_k)^T D_{k,t}(z-c_k)$$

Dies kann so interpretiert werden, dass sich bei einem Modell die Aktivierung in der Richtung zum Eingangsvektor ändert und in Richtungen, die (nach der Hauptachsentransformation) senkrecht zu der ausgewählten Richtung stehen, unverändert bleibt.

Simulationsbeispiel

Schaal & Atkeson [5] messen die Lernfähigkeit ihres Algorithmus, der sich ebenfalls mit räumlichen Aufteilungen befasst, anhand einer einfachen Funktion:

$$z = max\left(e^{-10x^2}, e^{-50y^2}, 1{,}25e^{-5(x^2+y^2)}\right) + N(0, 0{,}01)$$

Die Lernmenge besteht aus 500 Punkten, die gleichverteilt im Einheitsquadrat gezogen werden.

Die Güte der Anpassung wird an den Knotenpunkten eines 41x41-Gitters gemessen.

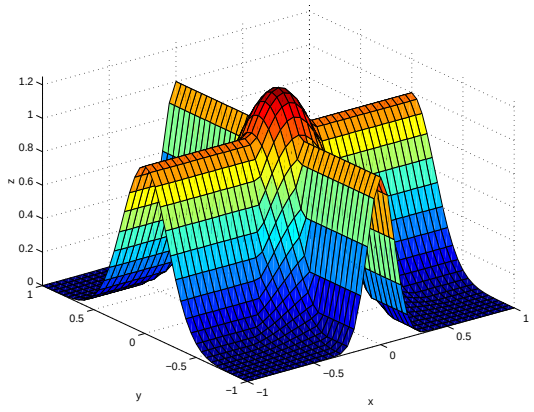


ABBILDUNG 8: Beispielfunktion von Schaal und Atkeson

Der Fehler wurde als der normierte MSE gemessen (MSE dividiert durch die Varianz der Ausgangsgröße).

Schaal & Atkeson erhalten einen Wert von $nMSE = 0{,}02$ nach 50 Epochen. Hierfür benötigten sie 48 Modelle. Ihnen gelang es auch nicht, mit verschiedenen MLPs mit 20 bis zu 100 Neuronen im Hidden Layer den Wert von $nMSE = 0{,}10$ zu unterschreiten. Ein „mixture of experts“ -System erzielte einen Fehler von $nMSE = 0{,}04$.

Bei unserem Verfahren verwendeten wir ebenfalls 48 lokale Modelle, deren Stützstellen zufällig über das Einheitsquadrat verteilt wurden. Die verwendeten Parameter waren $\eta = 0{,}01$, $e_{max} = 0{,}1$, $M = 5$ und $\rho = 0{,}01$. Die Matrizen D_k wurden bei der Initialisierung mit $60 \cdot I$ angesetzt (I sei die Einheitsmatrix) und der Vergessensfaktor war $\lambda = 0{,}999$.

Die Abb. 9 zeigt, wie sich der Fehler auf dem Testgitter mit der Zeit entwickelt. Nachdem die Trainingsdaten einmal gezeigt wurden, war der Fehler mit 0.08 deutlich geringer als die berichteten 0.12 von Schaal & Atkeson. In der Abb. 9 sieht man auch, wie sich der Fehler ohne das Update der Distanzmatrizen entwickelt, also mit ρ gleich null. Es wurden auch die gleichen Anfangspositionen der Stützstellen verwendet wie im ersten Fall. Werden die vorhandenen 500 Datensätze wiederholt zum Trainieren verwendet, sinkt der Fehler auf der Testmenge weiter, und für $\rho = 0{,}01$ ist nach nur 20 Epochen der von Schaal&Atkeson angegebene Wert von $nMSE = 0,02$ erreicht. Mit $\rho = 0$ fällt der Fehler nur auf $0{,}04$.
Die Abb. 10 zeigt die gute Approximation nach 20 Epochen. Betrachtet man den Contourplot gleicher Aktivitäten, fällt die Struktur der Ellipsen auf, die sehr gut zu der zu approximierenden Funktion passt.

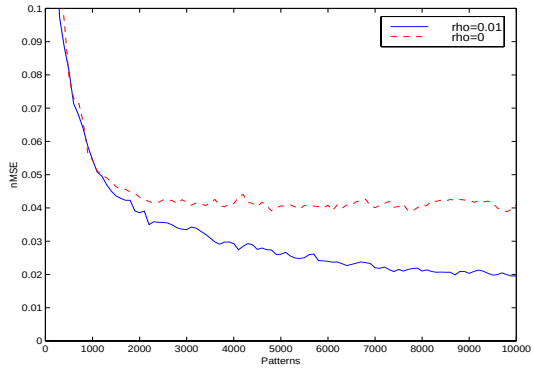


ABBILDUNG 9: Fehlerverlauf mit und ohne Anpassung der Aktivierungsfunktionen

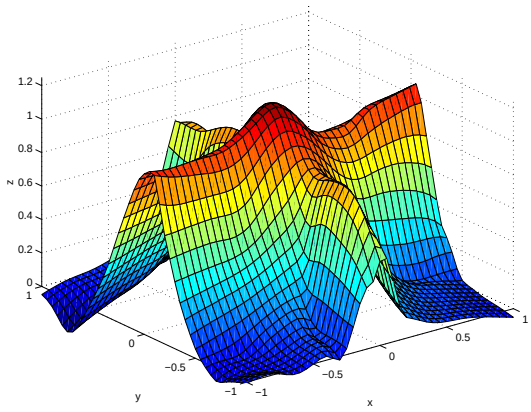


ABBILDUNG 10: Approximation nach 20 Epochen ($\rho = 0{,}01$)

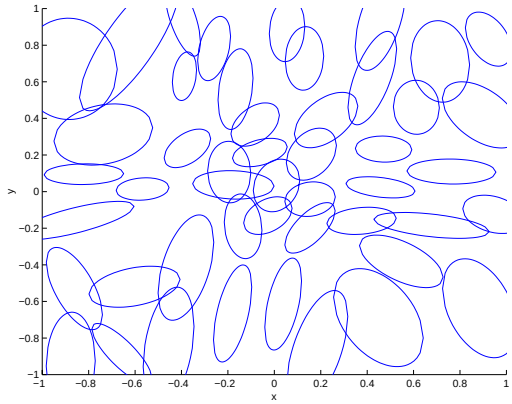


ABBILDUNG 11: Contour plot der Aktivierungen nach 20 Epochen ($\rho = 0{,}01$).

Eine Erweiterung

Zusätzlich zum dargestellten Grundmodell des letzten Abschnitts wurde noch eine Variante des Modellupdates eingeführt, die auf der Annahme basiert, dass sich die Zeitvarianz hauptsächlich in einer Änderung des Offsets ausdrückt. Bevor das eigentliche Modellupdate mittels einer gewichteten Regression durchgeführt wird, wird der Offset adaptiv angepasst. Dem bisherigen Schritt

$$\hat{\beta}_{t+1} = \hat{\beta}_t + ewP_{t+1}x$$

wird der Vorverarbeitungsschritt

$$\hat{\beta^o}t+1 = \hat{\beta^o}t + \tau \cdot \left(y - \hat{\beta^o}t\right)$$

vorangestellt, wobei nur die Offsetkomponente des Parametervektors verändert wird. Dieses Update wird auch nur für das Modell mit der höchsten Aktivierung durchgeführt.

2.4.3 Die Zeit als Eingangsgröße

Ein großer Teil aller industriellen Anwendungen von Neuronalen Netzen basiert noch immer auf Multilayer Perzeptrons. Dies ist vor allem in der Bekanntheit des Verfahrens und in der Vielzahl verfügbarer Tools begründet, die eine Implementation sehr einfach machen. Daher wurde in diesem Projekt auch untersucht, inwieweit die verschiedenen Problemstellungen mit diesem Standardnetz befriedigend gelöst werden können. Dabei fanden wir einfache Methoden, kontinuierliches Lernen sowie Fehler- und Vertrauensabschätzungen mit statischen unmodifizierten MLP's zu realisieren..

Kontinuierliches Lernen durch Entfalten der Zeitdimension

Eine seit langem in der Statistik angewandte Methode, v.a. bei Regressionsmodellen ist es, die Zeit als zusätzliche Eingangsgröße zu verwenden, wenn die Annahme besteht, dass der zu modellierende Zusammenhang nicht zeitinvariant ist. Dadurch wird ein n-dimensionales zeitvariantes Problem in eine n+1 dimensionale statische Form überführt. Auf Neuronale Netze übertragen bedeutet das, einfach die Zeit als zusätzlichen Eingang in die Eingangsschicht einzuführen (Abb. 12).

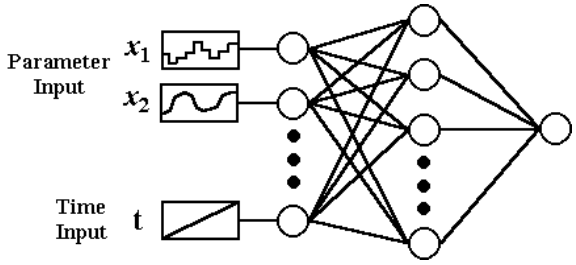


ABBILDUNG 12: Ein MLP wird durch einen zusätzlichen Eingang, auf den die Zeit abgebildet wird, zur Prognose von zeitveränderlichen Kennlinien befähigt.

Da die meisten kontinuierlich lernenden Verfahren dem Prozess nacheilen, jede Prognose aber eine Extrapolation in die Zukunft ist, liegt hier ein grundlegendes Problem aller Verfahren, die versuchen Zeitvariantes Verhalten aus Trainingsdaten zu lernen. Erst durch Erweiterungen um die Zeit explizit berücksichtigende Fehlerkorrekturen wie z.B. auf den Prognosefehler angewandte ARMA oder ARIMA Modelle, kann eine sinvolle Vorhersage von zukünftigen Zuständen gemacht werden. Die Netze mit Zeiteingang dagegen führen automatisch diese Extrapolation durch, wenn man den korrekten Prognosezeitpunkt auf diesen Eingang legt (Abb. 13).

Ein Nachteil liegt durch den zusätzlichen Eingang in einer erhöhten Komplexität des Modells, das dadurch deutlich rauschempfindlicher weden kann, besonders, wenn noch wenig Trainingsdaten vorliegen.

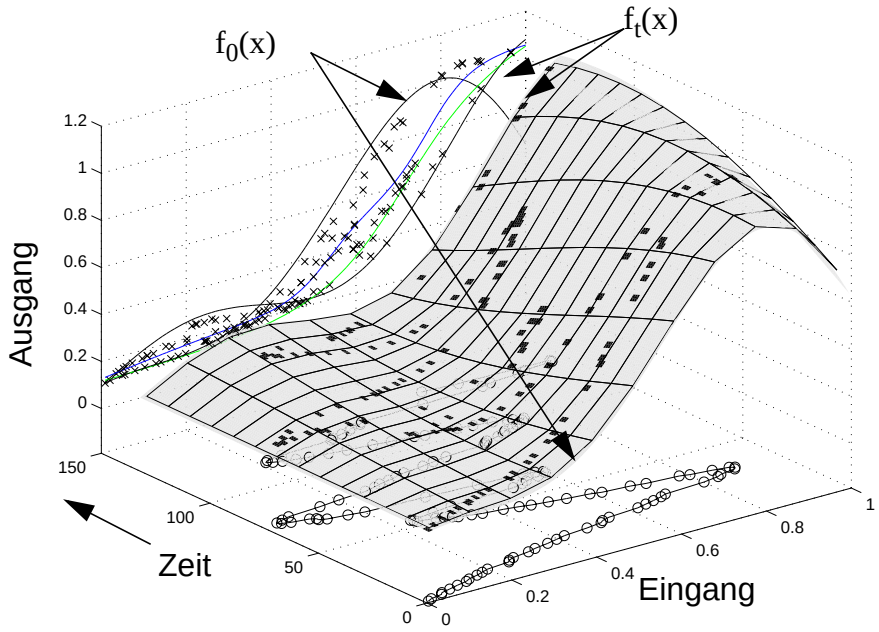


ABBILDUNG 13: Eine Kennlinie wird durch die nichtlineare Funktion $f(x)$ beschrieben. Allerdings ist diese zeitvariant, d.h. deren Form ändert sich kontinierlich mit der Zeit von $f_0(x)$ zu $f_t(x)$. Trainingsdaten (schwarze Punkte auf der Fläche) werden währenddessen aber nur an einzelnen Arbeitspunkten (beispielhafter Verlauf des Eingangs über der Zeit) gesammelt. Ziel zum Zeitpunkt t ist jedoch nicht nur die Vorhersage des nächsten Ausgangswertes (Punktvorhersage), sondern der kompletten Funktion $f_{t+1}(x)$ für den gesamten Wertebereich von x (Modellvorhersage). Lernverfahren, die die Zeitabhängigkeit nicht korrekt berücksichtigen, können diese Zielfunktion nur schlecht approximieren.

Mit einem Vergleich auf Benchmarkdaten konnten wir zeigen, dass die Modellierungsgüte dieser Methode in vielen Fällen anderen Verfahren deutlich überlegen ist. In einem Vergleich mit adaptiven lokalen linearen Karten und lokal gewichteter Regression lieferte das MLP mit Zeiteingang in 4 von 5 Benchmarks die besten Ergebnisse. [68]

2.4.4 Abschätzung der Vertrauenswürdigkeit Neuronaler Netze

Neuronale Netze als universale Funktionsapproximatoren liefern zu jeder beliebigen Kombination von Eingangsgrößen stets ein Resultat, unabhängig davon, wie erfolgreich das Training war. Auch wenn die aktuellen Eingangsgrößen in den vorhandenen Trainingsdaten überhaupt keine Repräsentation haben, wird das Netz ein beliebiges, bestenfalls zweifelhaftes Ergebnis anzeigen. Wir stellen einfache Methoden vor, die neben der Netzprognose eine parallele Ausgabe von zu erwartendem Fehler und der Vertrauenswürdigkeit liefern.

Motivation

Eines der Haupteinsatzgebiete von Neuronalen Netzen ist die datengetriebene Modellbildung (Identifikation) von nichtlinearen Prozessen, für die kein analytisches Modell zur Verfügung steht. Mit Hilfe eines solchen Neuro-Modells kann dann z.B. die Reaktion eines Prozesses auf verschiedene Stellgrößen simuliert und vorab eine optimale Stellgrößensequenz ermittelt werden (Modellbasierte Prädiktive Regelung). Offensichtlich hängt die Güte einer solchen Prozessoptimierung entscheidend von der Güte des Modells ab. Die Probleme und Lösungsansätze, die sich hierbei durch den „Black-Box Charakter“ von Neuronalen Netzen ergeben, werden im folgenden am Beispiel von Multilayer Perzeptrons erörtert.

Multilayer Perzeptrons sind in der Anwendung wegen ihrer Robustheit und einfachen Implementierbarkeit einer der meistverwendeten Netztypen. MLPs liefern für beliebige Werte an den Eingängen ein Ergebnis, ob sinnvoll oder nicht. Außerhalb des Arbeitsbereiches, für den Trainingsdaten vorliegen, wird diese Netzprognose in der Regel jedoch falsch sein. Im Gegensatz zu mathematisch - physikalischen Modellen, bei denen man normalerweise eine Plausibilitätskontrolle und Fehlerrechnung mit implementiert, fehlen entsprechende Kontrollen bei Neuronalen Modellen meist. Man beschränkt sich bei der Gütebewertung auf die Angabe des mittleren Fehlers auf dem Trainings-, Test- oder Validierungsset, was jedoch nichts über die Vertrauenswürdigkeit eines einzelnen Ausgabewerts für einen bestimmten Eingangsvektor aussagt. Fehlerhafte Netzprognosen können verschiedene Ursachen haben, darunter:

1. Der Ausgangswert ist prinzipiell nicht aus den vorhandenen Eingangsdaten vorhersagbar wegen verrauschter, fehlender oder widersprüchlicher Trainingsdaten.

2. Das Netz ist nicht korrekt trainiert oder ist in seiner Struktur nicht geeignet, die Daten zu repräsentieren.

3. Das vorhandene Trainingsdatenmaterial repräsentiert nicht die aktuelle Prognoseaufgabe, z.B. weil in den Trainingsdaten kein ähnlicher Fall vorhanden ist.

Alle diese Fehlerquellen können vom aktuellen Arbeitspunkt abhängen, Punkt drei mit Sicherheit. Also wäre es für den Anwender wünschenswert, Informationen über die zu erwartende Genauigkeit zu erhalten, die über den mittleren Fehler auf einem ganzen Datensatz hinausgehen. Für jeden Eingangswert sollte ein System nicht nur den modellierten Ausgangswert sondern auch ein Vertrauensmaß für die Gültigkeit/Verlässlichkeit berechnen können. Im Folgenden werden einfache Methoden vorgestellt, ein solches Vertrauensmaß für jeden Eingangswert zu ermitteln.

Vertrauenswürdigkeit von MLPs

Das Problem, den Fehler, den ein Modell macht, vorherzusagen, ist sehr ähnlich dem Problem das System selbst vorherzusagen. Eine Modellierung des vorzeichenbehafteten Fehlers ist natürlich nicht sinnvoll: Falls diese nicht Null ergäbe, wäre ja genausogut eine bessere Modellierung des Systems möglich. Allerdings wird auch ein optimal angepasstes Modell immer noch Fehler machen, die um Null herum verteilt sind. Dieser wird gewöhnlich absolut gemittelt über eine größere Testdatenmenge als Netzgüte angegeben. Um darüberhinausgehende lokalisierte Abschätzungen zu erhalten, wird ein zweites Netz, das die gleiche Struktur haben kann, mit den *gleichen Eingangswerten* aber dem absoluten oder quadratischen Fehler des Prognosenetzes als Zielfunktion trainiert. Um zu kleine Fehlerangaben durch Überanpassung auszuschließen, sollte dieses auf einem zweiten Validierungsdatensatz durchgeführt werden. Dieses Netz kann dann parallel zum ersten betrieben werden und liefert für jeden Eingangswert den mittleren zu erwartenden Fehler, abhängig von der Lage im Eingangsraum.

Allerdings versagt dieses Netz genauso für Eingangswerte, die außerhalb des Trainingsdatenbereiches liegen, da ja hier keinerlei Vorwissen vorhanden ist, auch nicht über die Fehler, die gemacht werden können. Diese dritte Fehlerquelle kann durch die Kenntnis der statistischen Verteilung der Eingangsdaten abgeschätzt werden [6]. Dieses kann ebenfalls durch ein weiteres, strukturell gleiches Netz erreicht werden. Dazu trainiert man es wiederum mit den *gleichen Eingängen*, jedoch mit einer Konstanten „1“ als Zielfunktion. Zusätzlich erzeugt man sich zufällige Eingänge aus dem gesamten zu erwartenden Arbeitsbereich und trainiert für diese mit dem Zielwert „0“. Dadurch ist dieses Netz in der Lage, die Verteilungsfunktion der Eingangswerte der Trainingsdaten zu modellieren: Es wird überall in Richtung Null gezogen, nur an Orten mit Trainingsdaten zur Eins hin. Durch Mittelung wird somit für jeden Bereich das Verhältnis von Zufallsvektoren und Trainingsdaten modelliert. Ein Ausgabewert nahe Null bedeutet also, dass man sich in unbekanntem Terrain bewegt und die Prognose des Prozessmodells nicht durch antrainiertes Vorwissen abgesichert ist. In den Abb. 14 - Abb. 18 wird dies anhand eines beispielhaften Funktionsverlaufes demonstriert [70].

Es hat sich als günstig herausgestellt, etwa 10-mal soviele Null-Vektoren zu erzeugen, wie Trainingsdaten vorhanden sind. Der Bereich sollte für jede Eingangsdimension klar über die maximal und minimal zu erwartenden Werte hinausgehen, um am Rand den Ausgang auf Null zu zwingen.

Anwendung bei der Optimierung von Prozesseinstellungen

Ein verbreiteter Anwendungsbereich für Neuronale Prozessmodelle ist die Optimierung von einstellbaren Parametern. Dabei werden durch Simulation am Modell die variierbaren Einstellungen gesucht, die bei gegebenen fixen Eingangsgrößen eine ebenfalls gegebene Zielgröße bestmöglich annähern oder den Ausgangswert maximieren oder minimieren. Dazu variiert ein Optimierungsalgorithmus die variablen Eingänge auf der Suche nach dem bestmöglichen Ausgang. Ein großes Problem dabei ist, dass bei Neuronalen Netzen Extremwerte oft weit außerhalb des Trainingsbereiches liegen. Benutzt man z.B. ein Gradientenabstiegsverfahren zur Einstellung der Eingänge, wird es oft auf Werte weit außerhalb des Bereiches konvergieren, der

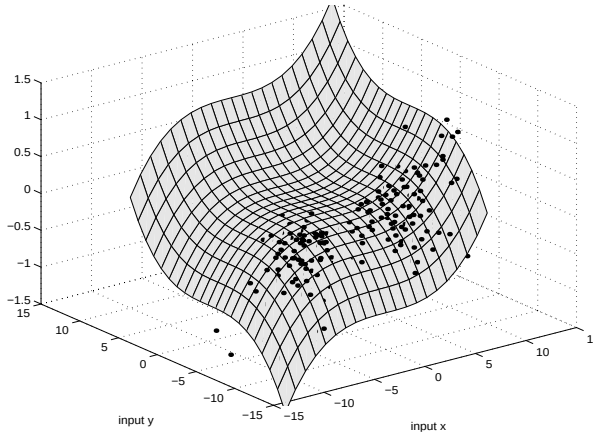


ABBILDUNG 14: Ein Kennfeld $z = f(x,y)$ wird an diskreten Punkten (x,y) abgetastet, die einer nicht gleichmäßigen Verteilung entstammen. Diese dienen dann als Trainingsdaten für ein MLP.

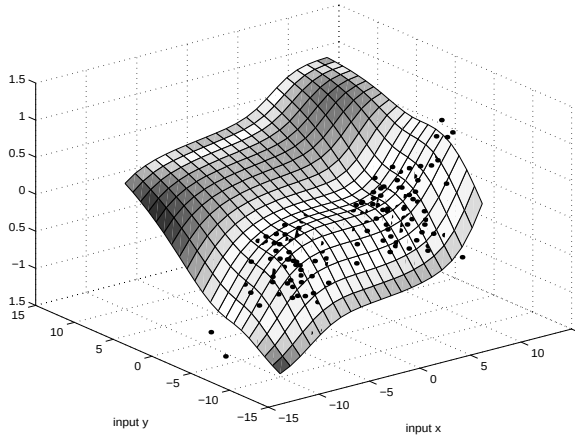


ABBILDUNG 15: Nachdem das Netz mit diesen Werten trainiert wurde, liefert es dieses Kennfeld *f'(x,y)*: In Gebieten mit vielen Trainingspunkten ist die Approximation genau (hell), in leeren Gebieten macht es Fehler (dunkel).

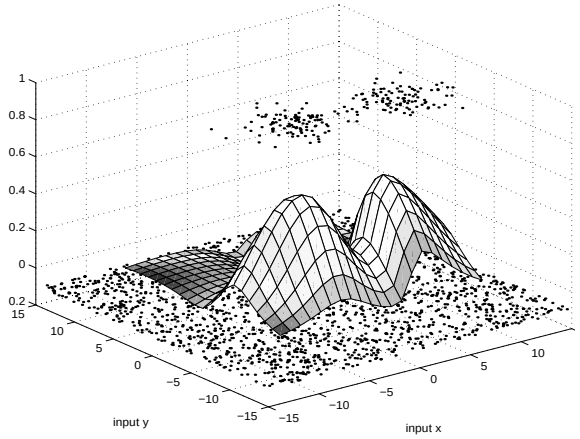


ABBILDUNG 16: Ein anderes Netz mit gleicher Topologie wurde mit den selben Eingangswerten *(x,y)* aber konstantem Target *z=1* sowie zufällig generierten Eingängen mit dem Target *z=0* trainiert. Dieses Netz hat gelernt, die Verteilung der Trainingspunkte im Eingangsraum abzubilden.

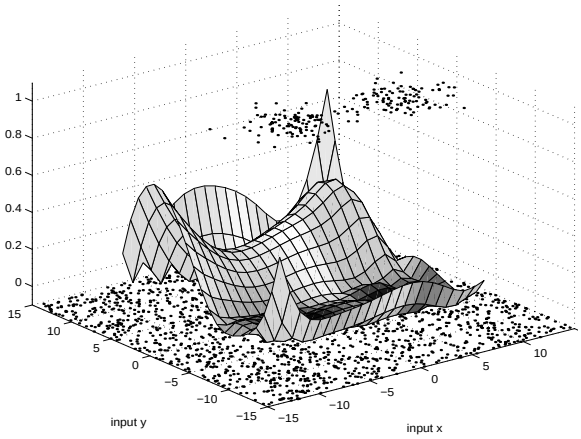


ABBILDUNG 17: Der Absolutfehler $|f - f'|$ der Kennlinienapproximation. Dunkle Regionen markieren Gebiete mit hoher Datendichte, wie sie das zweite Netz erkannt hat.

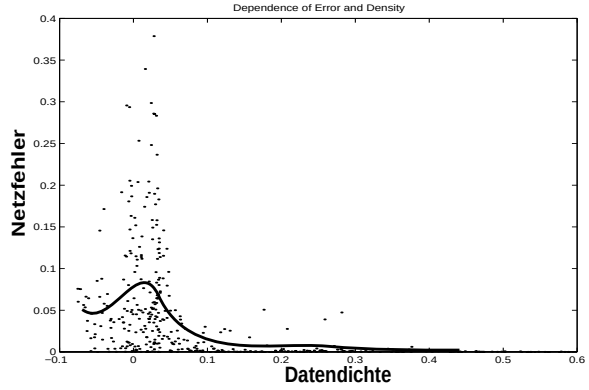


ABBILDUNG 18: Auf einem Testdatensatz erkennt man erkennt deutlich den Zusammenhang zwischen dem Vorhersagefehler für das Kennfeld an einem Punkt im Arbeitsraum und der Trainingsdatendichte an dieser Stelle. Das Dichteapproximationsnetz kann somit zur Berechnung einer Vertrauenskenngröße verwendet werden.

durch Trainingsdaten erfasst ist. Man muss manuell zusätzliche Randbedingungen und Einschränkungen einführen, um sinnvolle Ergebnisse zu erhalten. Durch die Einbeziehung der beschriebenen Fehlernetze in die Optimierung (Abb. 19) kann dieses Problem entschärft werden:

1. Es sind keine Schritte erlaubt, die in Bereiche führen, in denen die Ausgabe des „Verteilungsnetzes“ (NN3) auf nahe Null zurückgeht, also ins „Terra Incognita“. Dieses entspricht der automatischen Einschränkung auf den Bereich der vorhandenen Trainingsdaten.

2. Schritte in Richtungen, in denen zwar das Ergebnis des Modells besser wird, jedoch der Fehler stark steigt, werden schwächer berücksichtigt als Änderungen deren Auswirkung „sicher“ bekannt ist. Um die mittlere Abweichung vom Zielwert so klein wie möglich zu halten, muss man sogar die Summe aus quadratischer Abweichung von Ziel- und Istwert aus NN1 plus den erwarteten quadratischen Fehler aus NN2 minimieren.

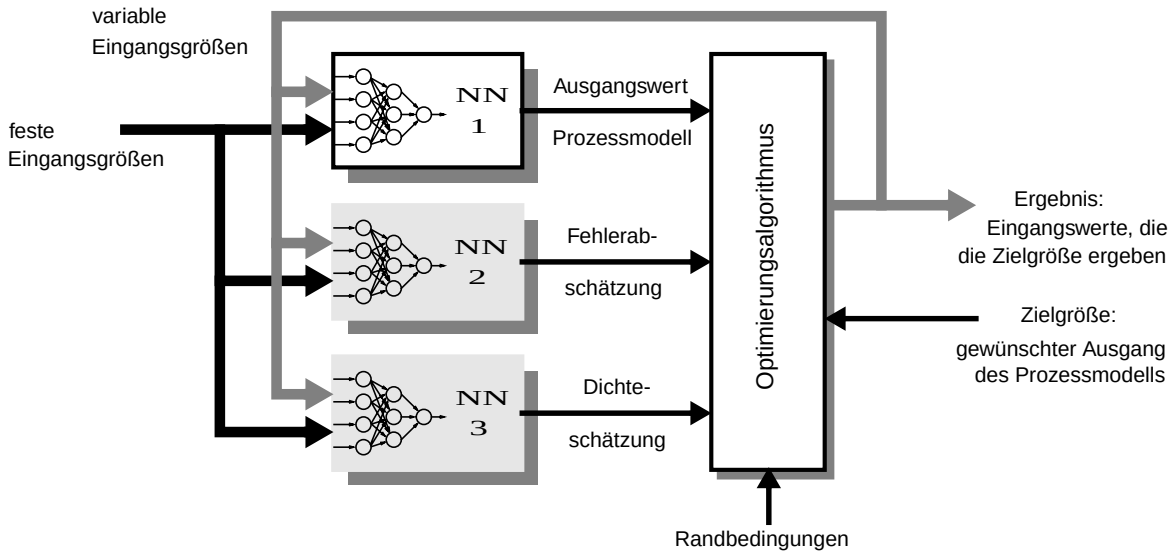


ABBILDUNG 19: Die Kombination von 3 strukturell gleichen Netzen ermöglicht einem Optimieralgorithmus bei seiner Suche nach optimalen variablen Parametern zur Erreichung einer Zielgröße, Einstellungen zu vermeiden, für die das Prozessmodell NN1 nicht vertrauenswürdig ist: Das Modell macht große Fehler (NN2) oder hat noch keine entsprechenden Trainingsdaten gesehen (NN3). Dazu wird NN2 mit dem Absolut- oder quadratischen Fehler von NN1 trainiert und NN3 zur Approximation der Datendichte im Eingangsraum der Trainingsdaten verwendet.

Einordnung

Die hier vorgeschlagene Methode zur Fehlermodellierung und Plausibilitätskontrolle von MLPs ermöglicht bei sehr geringem zusätzlichen Aufwand (man verwendet ja die gleiche Netzstruktur 3-fach parallel) häufig auftretende Probleme im industriellen Einsatz zu erkennen und zu vermeiden.

Bisherige Methoden zur Bestimmung von Dichten oder Verteilungen nutzen vor allem statistische Modellierungen durch Gaußsche Verteilungen [7][8] oder verwenden Baysian Netze [9][10]. Damit ist eine zwar eine exaktere Modellierung der Verteilungseigenschaften möglich (z.B. das Integral normiert sich automatisch zu Eins), der benötigte Aufwand übersteigt aber häufig den zur eigentlichen Funktionsapproximation.

Verwendet man zur Funktionsapproximation lokale Verfahren wie Radiale Basisfunktionsnetze oder Lokale Lineare Karten, kann man Vertrauenswürdigkeitsmaße auch direkt aus den Aktivierungen der jeweiligen lokalen Modelle ableiten, ohne dass man parallele Netze einführen muss [72].

3 Neue Adaptive Neuronale Netzarchitektur für das kontinuierliche Lernen und das Initiallernen

3.1 Anforderung an das Verfahren und Eignung verschiedener Modelle

Ein neues Lernverfahrens, dass sich für das kontinuierliche Lernen eignen soll, muss einige Rahmenbedingungen erfüllen. Es wird untersucht, welche dieser Anforderungen von bereits bestehenden Verfahren erfüllt bzw. verletzt werden. Die Idee des in diesem Abschnitt neu hergeleiteten Verfahrens besteht darin, die positiven Eigenschaften der betrachteten Verfahren zu extrahieren und die Vorteile zu vereinen.

Das neue Lernverfahren soll in der Lage sein, nicht nur ein allgemeines Modell des zu identifizierenden Prozesses abzuleiten, sondern darüber hinaus eine adäquate Netztopologie automatisch zu generieren. Als Lernstrategie wird eine progressive Vorgehensweise favorisiert, da bereits in der Initiallernphase plausible Netzausgaben zur Verfügung gestellt werden sollen.

Die Zeitvarianz, z.B. hervorgerufen durch Verschleiß und Alterung, erfordert eine schritthaltende Adaption an den sich verändernden Prozess (vgl. Abb. 13). Hierbei ist ein einmaliges Trainieren und anschließendes *Einfrieren* eines Modells nicht ausreichend, sondern es muss während der gesamten Lebenszeit des Prozesses kontinuierlich *nachtrainiert* werden können.

Eine sinnvolle Balance zwischen Stabilität und Plastizität zu finden, zählt zu den zentralen Problemen des Kontinuierlichen Lernens (Stabilitäts-Plastizitäts-Dilemma) [11]. In der Praxis bedeutet dies, so wenig wie möglich des bereits erlernten Wissens zu „verlernen“. Neben der Stabilität muss aber auch *Plastizität* gewährleistet sein, d.h. das Modell muss sich an lokale Veränderungen des Systemverhaltens anpassen können.

Obwohl Stabilität und Plastizität zwei widersprüchliche Eigenschaften sind, muss ein entsprechendes Lernverfahren in der Lage sein, für die jeweilige Situation einen „sinnvollen“ Mittelweg zu finden.

Eine weitere, wünschenswerte Eigenschaft ist eine Aussage über die Vertrauenswürdigkeit der generierten Netzausgaben (vgl. den Abschnitt über Vertrauenswürdigkeit auf S. 27). Idealerweise sollte das Lernverfahren selbst einen zusätzlichen Parameter für die Vertrauenswürdigkeit seiner Prognosen zur Verfügung stellen.

Ein Pluspunkt jedes Ansatzes ist es, wenn dem Benutzer zusätzlich zum abgeleiteten Prozessmodell Information über den unbekannten Prozess zur Verfügung gestellt werden kann.

Schliesslich sollte der hergeleitete Ansatz praktikabel sein und mit realistischen Datendimensionen umgehen können.

Multi-Layer-Perceptrons

Das wohl populärste und bekannteste neuronale Netzmodell ist das *Multi-Layer-Perceptron* (MLP) [12]. Einige der bekanntesten Lernalgorithmen sind Backpropagation (BP), Quickpropagation (QP) und Resilientpropagation (RPROP)[13][14]. Ein bemerkenswert Aspekt der MLPs ist, dass gezeigt werden kann, dass MLPs mit zwei verdeckten und trainierbaren Schichten universelle Approximatoren darstellen [15][16][17]. Den genannten Lernalgorithmen ist gemeinsam, dass in Abhängigkeit der Fehlerinformation am Ausgang die freien Parameter

ausgehend vom Ausgang rückwärts durch das Netz adaptiert werden. Für Kontinuierliches Lernen ist jedoch die Tatsache, dass das Fehlersignal durch das *gesamte* Netz propagiert wird, von grosser Bedeutung. Dies hat zur Folge, dass jede lokale Beobachtung globale Auswirkungen auf das bereits erlernte Modell haben kann. Verstärkt wird dies durch die globale Ausdehnung der Aktivierungsfunktionen in den einzelnen Neuronen. Diese Eigenschaft *globaler Modelle* kann zu großen Problemen führen [18]. Plastizität kann z.B. gewährleistet werden, indem dem Lernverfahren nur die lokale zeitliche Vergangenheit präsentiert wird [11]. Dies wird über ein gleitendes Datenfenster mit begrenzter Länge erreicht. Lernalgorithmen wie BP sind dadurch in der Lage, das identifizierte Modell an die aktuelle Situation anzupassen. Eine Schwierigkeit besteht jedoch darin, die optimale Länge des Fensters zu finden. In Abschnitt 2.4.3 wurde eine Methode dargelegt, diese Schwächen von MLPs durch die Entfaltung der Zeitdimension zu umgehen.

MLPs werden sinnvollerweise in solchen Fällen eingesetzt, in denen kein bzw. kein ausreichendes Streckenmodell des zu steuernden Prozesses vorhanden ist. Da MLPs sich nach außen wie schwarze, undurchsichtige Kästen (black boxes) verhalten, ist es äußerst schwierig, auf der Basis des abgeleiteten Prozessmodells Informationen über den unbekannten Prozess zu extrahieren. Ursache ist, wie bereits bei der Vertrauenswürdigkeit der Netzausgabe, die fehlende Transparenz aufgrund der subsymbolischen, verteilten Wissensrepräsentation.

Radiale-Basis-Funktions-Netze

Ähnlich wie Multi-Layer-Perceptrons sind Neuronale Netze mit Radialen-Basis-Funktionen universelle Approximatoren (RBF) [19][20]. Im Gegensatz zu MLPs werden bei RBF-Netzen anstelle sigmoider oder linearer Aktivierungsfunktionen radial-symmetrische Aktivierungsfunktionen, wie beispielsweise die Gauss-Funktion, verwendet. Diese in der Regel zweischichtige neuronale Netzform[1] bietet mehrere Vorteile gegenüber MLPs.

Ein Vorteil liegt im einfachen und sehr schnellen Training der Netze. Im einfachsten Fall werden immer dann RBF-Neuronen eingefügt, wenn der Fehler am Ausgang zu groß ist. Die RBF-Neuronen werden hierzu mit fester Ausdehnung an der aktuellen Eingabemusterposition eingefügt. Lediglich die gewichtete Verbindung zum Ausgabeneuron (lineare Aktivierungsfunktion) muss entsprechend des aktuellen Ausgabewerts berechnet werden. Der einfache Lernalgorithmus ist allerdings auch eine Schwachstelle bei industriellen Anwendungen. Der Grund ist, dass bei industriellen Anwendungen die Dimension des Eingaberaums sehr hoch sein kann. Um eine adäquate Approximation des nichtlinearen Prozesses zu erreichen, wird eine enorme Anzahl von RBF-Neuronen benötigt. Diese können in der Regel mit den zur Verfügung stehenden Ressourcen nur unzureichend verarbeitet werden.

Im Hinblick auf Kontinuierliches Lernen bieten RBF-Neuronen jedoch mehrere Vorteile. So wirken sich lokale Veränderungen nur lokal im Netz aus. Das heißt, nur das für den aktuellen Eingabebereich aktive Neuron muss adaptiert werden (z.B. Positionsverschiebung oder Aktivierungsanpassung). Ein weiterer Vorteil sind die radial-symmetrischen Aktivierungsfunktionen. Mit zunehmender Entfernung zum Zentrum fällt die Aktivierung auf nahezu Null ab. Das

1. Lediglich in der verdeckten Schicht werden RBF-Neuronen eingesetzt. Die Ausgabeschicht dient in erster Linie der Zusammenführung der einzelnen RBF-Neuronenausgaben.

heißt, dass die Aktivierung eines Neurons nur in einem lokalen, definierbaren Bereich signifikant ungleich Null ist und somit global nur einen minimalen Einfluss besitzt. Der „aktive“ Bereich wird auch als *rezeptives Feld* bezeichnet. Die rezeptiven Felder haben einen weiteren positiven Aspekt: Sie können als Indiz für die Vertrauenswürdigkeit der Netwerkausgabe eingesetzt werden. Genauer gesagt deuten Bereiche mit niedriger Dichte von RBF-Neuronen auf Eingaberaumgebiete hin, in denen nur wenig bzw. keine Trainingsdaten präsentiert wurden. Allerdings kann das nur als Indiz gewertet werden, da Gebiete mit sehr hoher Dichte auch auf starke Nichtlinearitäten des Prozesses hindeuten können. Was die Interpretrierbarkeit des abgeleiteten Prozessmodells angeht, gibt es sowohl positive als auch negative Aspekte. Positiv wirkt sich die relativ einfache Netztopologie aus. Negativ ist hingegen die hohe Anzahl an Neuronen, die eine sinnvolle Analyse erschweren oder verhindern.

Prinzipiell stellen RBF-Netze jedoch ein Werkzeug dar, das sowohl Stabilitäts- und Plastizitätsanforderungen erfüllt und darüber hinaus einen Hinweis auf die Vertrauenswürdigkeit der Netzausgabe gibt. Es besteht jedoch das Problem der hohen Neuronenanzahl. Möglichkeiten, wie die Anzahl der Neuronen reduziert werden kann, werden im Abschnitt 3.2 für den ICE-Algorithmus vorgestellt.

Einsatzmöglichkeiten sonstiger Verfahren

Ähnlich wie bei RBF-Netzen werden bei konstruktiven Verfahren, wie beispielsweise dem Cascade-Correlation-Algorithmus, immer neue Elemente in das Netz aufgenommen, um aktuelle Fehler zu minimieren [21][22]. Bei der Identifikation zeitvarianter Prozesse werden bei derartigen Ansätzen permanent neue Neuronen eingefügt. Dies ist mit einem ständig steigenden Speicher- und Berechnungsaufwand verbunden. Zwar existiert eine Reihe unterschiedlicher Pruning-Ansätze[1], bei zeitvarianten Prozessen ist es jedoch schwierig zu entscheiden, welche Neuronen oder Verbindungen entfernt werden können. Unter Umständen können somit Konflikte zwischen „alten“ und „neuen“ Neuronen entstehen. Eine Interpretation der komplexen Netze ist nahezu unmöglich. Ein Indiz für die Vertrauenswürdigkeit ist ebenfalls nicht abzuleiten. Das heißt, konstruktive Verfahren wie das Cascade-Correlation-Verfahren sind für Kontinuierliche Lernansätze ungeeignet.

Je mehr „konventionelle“ Verfahren wie MLPs oder RBF-Netze an Grenzen stoßen, desto interessanter werden so genannte Hybrid-Architekturen. Hierbei handelt es sich um Mischformen unterschiedlicher Ansätze. Besonders günstige Eigenschaften in Bezug auf Kontinuierliches Lernen besitzen zweischichtige Architekturen auf der Basis lokaler Modelle. Die erste Schicht ist für die „sinnvolle“ Aufteilung (clustering) des Eingaberaums verantwortlich. Die zweite Schicht übernimmt mit Hilfe lokaler Modelle die eigentliche Approximation des nichtlinearen Prozesses. Vertreter dieser Art von Verfahren sind beispielsweise Counter-Propagation-Netze [23]. Das Clustern des Eingaberaums erfolgt über eine Kohonenschicht, die unüberwacht trainiert wird [24]. Anschließend werden in der zweiten Schicht lokale lineare Modelle in den einzelnen Clustern angepasst.

1. Unter Pruning wird die Eliminierung vorhandener Netzelemente, wie Neuronen oder Verbindungen zwischen Neuronen, verstanden.

Diese hybride Form bietet für das Kontinuierliche Lernen einige Vorteile. So sind sowohl Stabilität als auch Plastizität durch die lokalen Modelle realisierbar. Lokale Beobachtungen wirken sich auch nur lokal aus, im einfachsten Fall nur auf das aktive Modell. Über die Kohonenschicht können Rückschlüsse auf die Trainingsdatendichte gezogen werden, die wiederum als Indiz für eine mögliche Vertrauenswürdigkeit der aktuellen Prognose dienen kann. Darüber hinaus lassen sich *Selbst-Organisierende-Karten* oder *Self-Organizing-Maps* (SOM), wie Kohonennetze auch genannt werden, auf einfache Art als 2D-Karten visualisieren. SOMs sind ein einfaches Werkzeug, mit dem der Prozessverlauf überwacht werden kann.

Neben diesen positiven Eigenschaften haben derartige Ansätze auch einige Nachteile. So können die lokalen Modelle erst angepasst werden, nachdem die Arbeitsraumaufteilung weitgehend abgeschlossen ist (Initiallernphase). Darüber hinaus benötigen SOMs in der Regel sehr viele Trainingsdaten. Für die Produktion bedeutet dies, dass solange mindere Qualität produziert wird, bis das Initiallernen abgeschlossen ist. Idealerweise sollten schon während der Initiallernphase sinnvolle Modellprognosen zur Verfügung stehen. Ein weiterer Nachteil ist die fehlende Kopplung bzw. Interaktion zwischen Modell- und Clusterebene. Dadurch kann es passieren, dass in einem Cluster das entsprechende Modell nicht in der Lage ist, die lokale Nichtlinearität zu modellieren.

Einen anderen Weg gehen Jacobs und Jordan mit dem Mixture of Experts Ansatz [25][26][27]. Hierbei werden globale Experten durch ein s.g. *Gating*-Netz lokal ein- bzw. ausgeblendet. Eine positive Eigenschaft ist, dass sich die einzelnen Experten in Abhängigkeit ihrer Fähigkeiten spezialisieren und somit die gesamte Performanz steigern. Nachteile sind, dass die komplexen Experten nur unzureichend interpretiert werden können und darüber hinaus ihre Struktur vorgegeben werden muss. Schnelles Initiallernen wird dadurch behindert.

Eine einfache Strategie zur automatischen Generierung einer adäquaten Netztopologie stellt der RCE-Ansatz dar [28]. Wenn Prognoseabweichungen größer als ein vorgegebener Schwellwert sind, wird das aktuelle Netz um neue Komponenten erweitert. Somit entfällt die aufwändige Ermittlung einer passenden Netzstruktur. Zu beachten ist, dass der RCE-Ansatz lediglich für binäre Klassifikationen entwickelt wurde.

3.2 ICE - Algorithmus

ICE steht für „*Incremental Clustering and Evolving/Evaluation*“. Dieser neuartige Ansatz ist die Kombination aus einem Hybrid-Netz und einem inkrementellen konstruktiven Lernalgorithmus. Die Basis des dreischichtigen Netzs bildet eine Clusterebene aus adaptiven RBF-Neuronen (vgl. Abb. 20). Die Zielfunktion wird durch lokale Modelle in der zweiten Ebene approximiert. Die Aufgabe des Ausgabeneurons in der dritten Ebene ist die Bereitstellung bzw. Berechnung der Netzprognose. Bemerkenswert ist, dass die Anzahl der RBF-Neuronen, ihre Position, die Ausdehnung der rezeptiven Felder sowie die Anzahl der lokalen Modelle ohne Vorgaben von Außen kontinuierlich während der Datenpräsentationsphase automatisch ermittelt werden.

Während über die lokalen Modelle (inkl. ihrer RBF-Neuronen) der Forderung nach Stabilität und Plastizität Rechnung getragen wird, dienen Position und Aktivierung der RBF-Neuronen als Indizien für die Trainingsdatenverteilung und einen Vertrauenswürdigkeitsfaktor für die

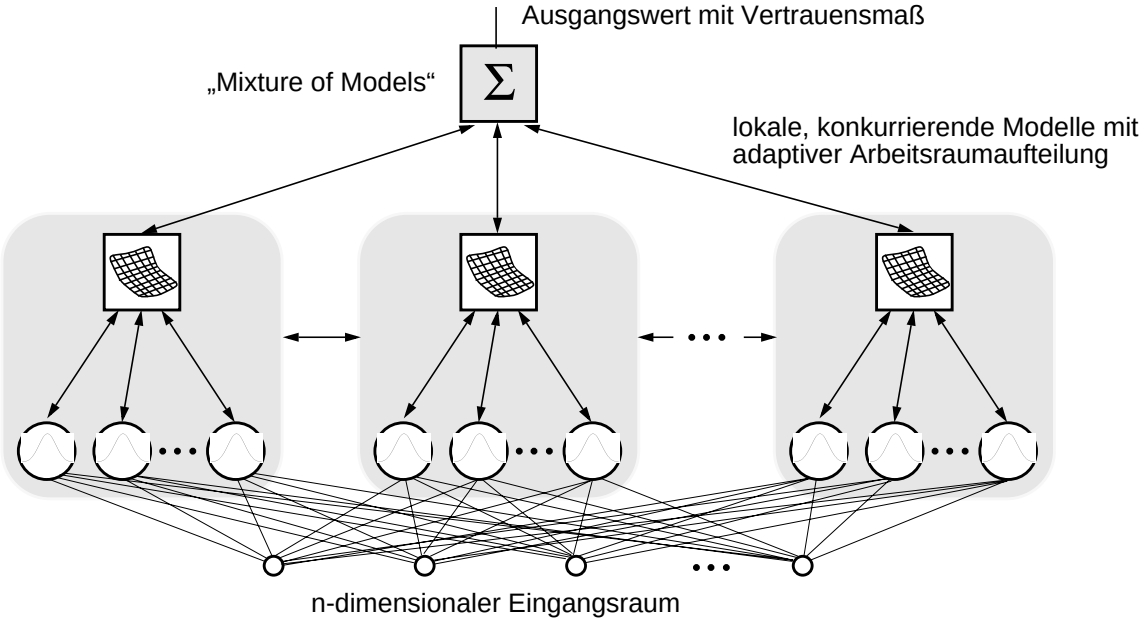


ABBILDUNG 20: ICE-Netzarchitektur. Die erste Ebene besteht aus spezialisierten RBF-Neuronen und dient der adaptiven Aufteilung des Eingaberaums. Die Approximation der Zielfunktion erfolgt in der zweiten Netzebene auf der Basis lokaler, konkurrierender Modelle.

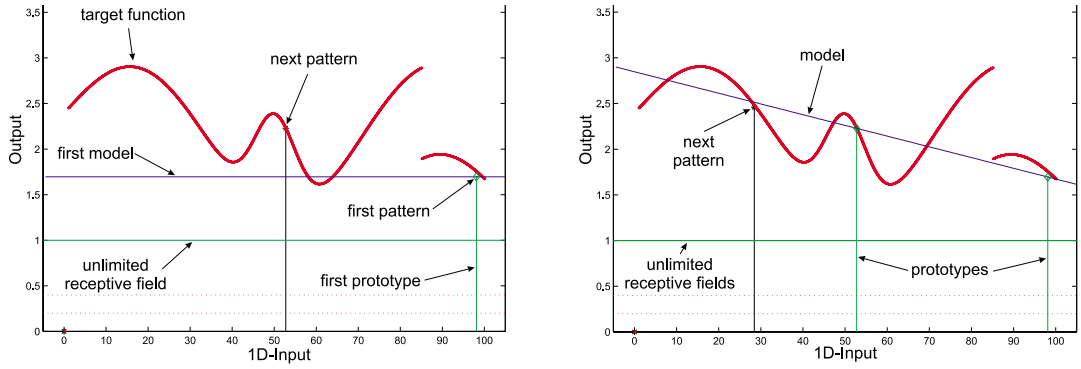
Netzprognose. Durch starke Interaktion zwischen den einzelnen Schichten stehen bereits in der Initiallernphase plausible Netzprognosen zur Verfügung. Wie schon angesprochen ist dies ein sehr wichtiger Aspekt für den industriellen Einsatz. Darüber hinaus wird eine Regelextraktion auf der Basis der Modelle angestrebt.

Adaptiver Hybrid Ansatz

Die Basis des ICE-Algorithmuses ist die Eigenschaft, dass sämtliche Modelle in *Konkurrenz* zueinander stehen. Dieser Wettbewerb bewirkt ein gegenseitiges Einschränken der rezeptiven Felder in der Cluster-Ebene. Jedem Modell sind mehrere spezialisierte RBF-Neuronen zugeordnet. Diese RBF-Neuronen werden im folgenden auch als *Prototypen* bezeichnet. Spezialisiert heißt in diesem Fall, dass diese Prototypen gegenüber konventionellen RBF-Neuronen über zusätzliche Eigenschaften verfügen. Ein Beispiel ist ein Offset-Parameter im Zentrum des Prototypen. Aufgabe dieses Parameters ist, als Stützstelle für das lokale Modell zu dienen. Wie schon angesprochen, können jedem Modell ein oder mehrere Prototypen zugeordnet werden. Dies hat den Vorteil, dass dadurch eine detailliertere Arbeitsraumaufteilung möglich ist. Mit dem Effekt, dass dadurch das Potential gegeben ist, die Anzahl der benötigten Modelle zu reduzieren. Da die Modellschicht als Basis für spätere Regelextraktionen dient, sollte diese so einfach wie möglich gestaltet werden. Im Fall linearer Modell ist beispielsweise der Einsatz von $n + 1$ Prototypen sinnvoll, wobei n der Dimension des Eingaberaums entspricht. In diesem Fall kann das lokale Modell schnell und exakt an seine jeweiligen Prototypen angepasst werden. Wie der Lernprozess genau abläuft, wird im folgenden anhand eines einfachen eindimensionalen Beispiels erläutert.

Erste Netzkomponenten

Das Training beginnt mit einem „leeren“ Netz. Leer bedeutet, dass kein Modell und dementsprechend keine Prototypen vorhanden sind. Somit kann und wird für das erste Trainingsmuster keine Prognose ausgegeben. In Abhängigkeit des ersten Trainingsmusters [$\boldsymbol{x};y$] ([Eingabevektor, Zielausgabe]) wird der erste Prototyp in das Netz integriert. Als Prototypzentrum wird die Position des Eingabevektors verwendet. Der bereits erwähnte Offset wird auf den Wert der Zielausgabe gesetzt. Anschließend wird auf diese „Stützstelle“ das erste Modell angepasst, so dass es konstant den ersten Zielwert (=Prototyp-Offset) ausgibt (siehe Abb. 21a). Mehr Information ist zu diesem Zeitpunkt auch nicht vorhanden.



(a) Nach erstem Trainingsmuster (b) Nach zweitem Trainingsmuster

ABBILDUNG 21: In (a) ist das Netz (erster Prototyp mit unendlicher Ausdehnung des rezeptiven Feldes und erstes Modell) nach der Präsentation des ersten Trainingsmusters dargestellt. (b) zeigt das adaptierte Modell nach der Präsentation des zweiten Trainingsmusters.

Abschließend muss die Ausdehnung des rezeptiven Feldes eingestellt werden. Hierfür sind unterschiedliche Strategien möglich. Im vorgestellten Ansatz wird eine progressive Strategie bevorzugt. Progressiv heißt, dass die rezeptiven Felder maximale Ausdehnung einnehmen. Die Grenzen werden nur durch konkurrierende Prototypen eingeschränkt. Für den ersten Prototypen folgt daraus, dass aufgrund fehlender Konkurrenz das rezeptive Feld unbegrenzt ausgedehnt werden kann.

Während der Präsentation neuer Trainingsmuster werden in Abhängigkeit der Prognosegüte neue Prototypen und Modelle erzeugt. Die Prototypzentren und der Offsetparameter werden wie beim ersten Prototyp berechnet. Das Modell wird direkt auf die Prototypen „aufgesetzt“ (vgl. Abb. 21b).

Adaption der Netzkomponenten

Solange die Prognosegüte ausreichend ist, werden keine neuen Modelle bzw. Prototypen erzeugt. Stattdessen wird der aktive Prototyp (Prototyp mit höchster Aktivierung) an die neuen Trainingsdaten angepasst. Da die Modelle die Prototypen als Stützstellen verwenden, werden auch die Modelle an die neuen Daten adaptiert.

Die Adaption des aktiven Prototypen an das aktuelle Trainingsmuster erfolgt dadurch, dass der Prototyp in Richtung des aktuellen Trainingsmusters bewegt wird. In Anlehnung an Kohonennetze wird hierzu das Prototypzentrum in Richtung des Eingabevektors und der Offset entspre-

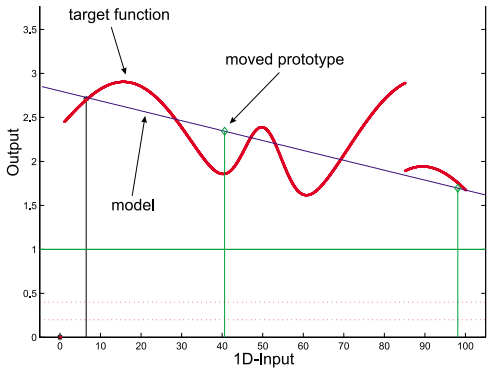
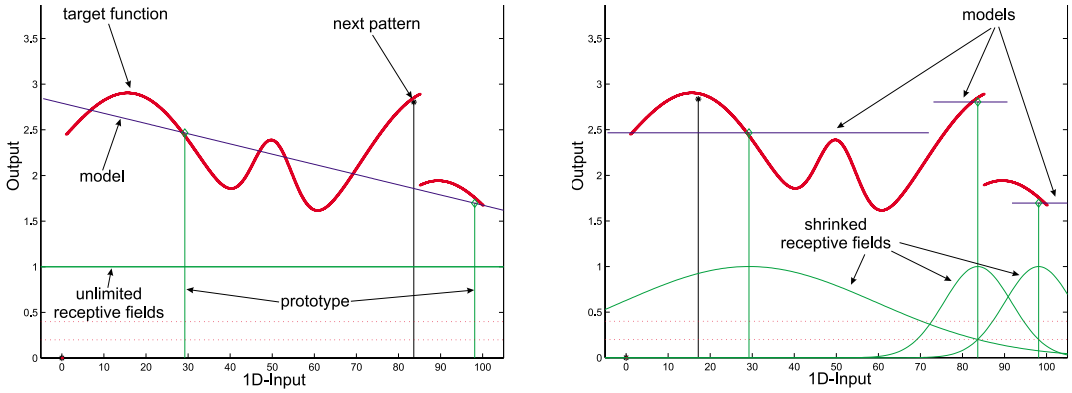


ABBILDUNG 22: Der aktive Prototyp (Zentrum und Offset) wird in Richtung des aktuellen Trainingsmusters (vgl. *next pattern* in Abb. 21b) verschoben .

chend an den Zielwert adaptiert. Die Verschiebung in Richtung des aktuellen Trainingsmusters kann auf unterschiedliche Arten gelöst werden. Eine Möglichkeit ist, den Prototypen immer im gleichen Verhältnis auf das aktuelle Trainingsmuster zu zu bewegen. Eine weitere Möglichkeit ist die Adaption in Abhängigkeit des Prototypalters (Alter ~ Anzahl der lokalen Beobachtung) auszuführen. „Junge“ Prototypen werden stärker, „ältere bzw. erfahrenere“ Prototypen werden nur wenig an das Trainingsmuster adaptiert. Wie weit der Bewegungsspielraum für ältere Prototypen eingeschränkt wird, hängt von der jeweiligen Anwendung ab. Ist der Prozess nahezu stationär, ist es sinnvoll, die Prototypen im Laufe der Zeit einzufrieren. Bei starker Zeitvarianz hingegen müssen die Modelle und dementsprechend die Prototypen in der Lage sein, ausreichend schnell auf die Veränderungen zu reagieren.

Splitten und Einschränken

Je nach Nichtlinearität der Zielfunktion und angestrebter Approximationsgüte kann es im Laufe der Zeit notwendig sein, neue Modelle zu erzeugen. Ein einfaches Beispiel ist in Abb. 23a dargestellt.



(a) Vor splitten (b) Nach splitten

ABBILDUNG 23: Da die Prognose für das aktuelle Eingabemuster (next pattern) nicht ausreichend ist, muss ein neues Modell erzeugt werden. Das aktive Modell wird aufgesplittet, der erste Prototyp des neuen Modells wird eingefügt, um anschließend die konkurrierenden rezeptiven Felder einzustellen. Abschließend werden die Modelle adaptiert.

Deutlich ist die Diskrepanz zwischen Netzprognose und angestrebter Zielausgabe zu erkennen. Da das aktive Modell die notwendige Datenapproximation alleine nicht leisten kann, ist es notwendig, neue Modelle in das System aufzunehmen. Bevor jedoch neue Modelle und entsprechend neue Prototypen integriert werden können, müssen einige Fragen beantwortet werden: Was geschieht mit existierenden Modellen? Wo müssen bzw. wo können existierende Modelle aufgesplittet werden? Und wie können die rezeptiven Felder sinnvoll eingeschränkt werden?

Es ist offensichtlich, dass das aktive Modell aufgebrochen werden muss. Die Frage ist nur, ob es immer notwendig ist, das aktive Modell in „alle“ Einzelteile zu zerlegen. Dies kann verneint werden. Nur wenn das aktuelle Trainingsmuster innerhalb der konvexen Hülle der aktiven Modellprototypen liegt, ist ein Aufsplitten nötig.

In niederdimensionalen Räumen ist die Frage, ob ein Punkt in der konvexen Hülle mehrerer Punkte liegt, recht einfach zu beantworten. In hochdimensionalen Vektorräumen ist dies mit enormem Rechenaufwand verbunden und für die Praxis inakzeptabel. Im Falle des ICE-Algorithmuses lässt sich das Problem jedoch sehr einfach lösen. In einem ersten Schritt wird das aktive Modell aufgesplittet, d.h. jeder Prototyp wird eigenständig. Danach wird an der Position des aktuellen Trainingsmusters ein neuer Prototyp eingefügt. Im nächsten Schritt werden die rezeptiven Felder gegen diesen neuen Prototypen eingeschränkt.

In Anlehnung an den Dynamic-Decay-Adjustment-Algorithmus (DDA) ist die Idee, dass Prototypen an der Position eines konkurrierenden Modellprototyps höchstens eine maximale Aktivierung Θ^- erreichen dürfen [29]. Auf der Basis dieses Kriteriums lassen sich nun alle rezeptiven Feldausdehnungen direkt berechnen. Beachtet werden muss, dass nur rezeptive Felder konkurrierender Modelle gegeneinander eingeschränkt werden und nicht Prototypen, die beispielsweise vor dem Modellsplitten in einem Modell verbunden waren. Im nächsten Schritt können alle Prototypen zu einem Modell zusammengefasst werden, die an der Position eines anderen Prototypen eine Aktivierung größer als Θ^- besitzen, da sie in diesem Fall nicht in Konkurrenz zueinander stehen können. Abschließend können nun die Modelle entsprechend ihrer lokalen Prototypen adaptiert werden (siehe Abb. 23b). Diese Vorgehensweise ist auch in hochdimensionalen Eingaberäumen ($\text{Dimension} \geq 20$) schnell und einfach zu berechnen[1].

Damit sind die wesentlichen Eigenschaften des ICE-Algorithmuses eingeführt. Im folgenden werden einige Ergebnisse für eindimensionale (1D), 2D und 11D Experimente vorgestellt[2].

3.3 Anwendungsbeispiele und Diskussion

In diesem Abschnitt werden einige Beispiele präsentiert, die das Lernverhalten und die Form der resultierenden Netze veranschaulichen sollen. Zu diesem Zweck werden zunächst Resultate auf 1D- und 2D-Daten vorgestellt, wobei die Zielfunktionen nicht zeitvariant sind. Anschließend folgt ein Beispiel einer zeitvarianten 1D-Funktion, gefolgt von 11D-Benchmark Ergebnissen. Abgeschlossen wird der Abschnitt mit der Präsentation auf 1D- und 2D-Trainingsdaten, bei

1. Beachtet werden muss, dass zum einen alle Fälle abgedeckt werden, in denen das neue Muster in der konvexen Hülle liegt, zum anderen aber auch Prototypen eines Modells getrennt werden können, in denen das Muster außerhalb der konvexen Hülle liegt. Allerdings nur dann, wenn mehr als *n+1* Prototypen pro Modell verwendet werden, wobei *n = Dimension des Eingaberaums* ist.
2. Die angegebene Dimension bezieht sich auf die Dimension des Eingaberaums.

denen die Daten in nicht gleichverteilter Form dem Netz präsentiert wurden. Die Daten der 1D- und 2D-Beispiele wurden alle mit 10% Rauschen gestört. Die 11D-Daten stammen von einem Projekt interner Benchmarks (vgl. Acknowledgement).

Statische Funktionen

Im ersten Beispiel wurden 200 verrauschte Muster einer nichtlinearen unstetigen Funktion dem Netz präsentiert. Erste Lernschritte wurden bereits in den Abbildungen 21, 22 und 23 vorgestellt. In Abb. 24 ist das Ergebnis nach einem Trainingsdurchlauf (Epoche) dargestellt.

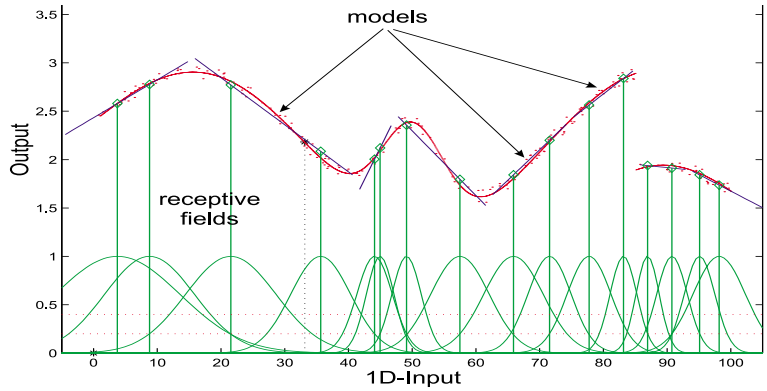
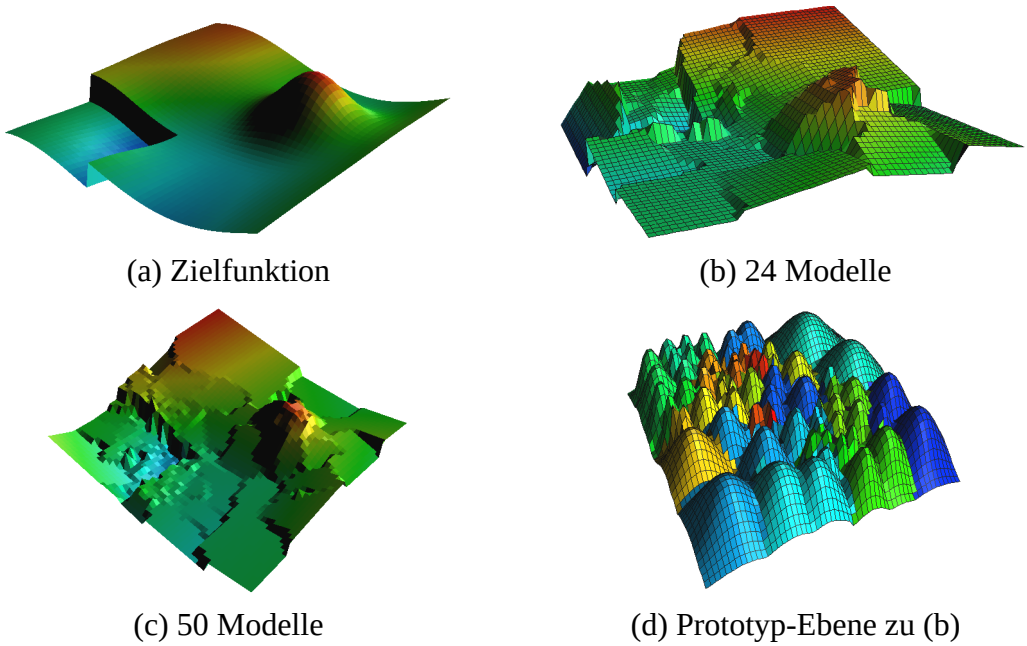


ABBILDUNG 24: Resultat nach einmaliger Präsentation von 200 verrauschten Trainingsmustern einer nichtlinearen und unstetigen Funktion.

Wie zu erwarten war, decken die Prototypen den gesamten Eingabebereich ab, das heißt den Bereich, in dem das Netz Daten beobachten konnte. Da die Prototypen nicht nur von den Eingabedaten, sondern darüber hinaus vom lokalen Modellfehler abhängen, wurden kleine rezeptive Felder in Bereichen starker Nichtlinearität und dementsprechend auch im Bereich der Unstetigkeitsstelle generiert. Vergleichbare Ergebnisse konnten auch in Versuchen mit 2D-Eingabedaten beobachtet werden (vgl. Abb. 25).

Auch in diesem Beispiel wurden gute Approximationen der nichtlinearen unstetigen Funktion gelernt. Wiederum wurden in Bereichen starker Nichtlinearität und im Bereich der Unstetigkeitsstelle kleinere rezeptive Felder erzeugt (siehe Abb. 25d). In beiden Versuchen wurden die gleichen Datensätze verwendet. Lediglich der adaptive Spielraum der Prototypen, respektive der erlaubte lokale Modellfehler, war im Experiment von Abb. 25c etwa halb so groß wie im Experiment aus Abb. 25b. Dementsprechend waren mehr Modelle für die Approximation nötig. In allen bisherigen Beispielen wurde sowohl in der Trainingsphase als auch in Anwendungsphase nur die Modellinformation des lokalen, aktiven Modells verwendet. Im Gegensatz hierzu wurde im Experiment aus Abb. 26 sowohl beim Training als auch in der Anwendungsphase Information aus allen Modellen kombiniert.

Hierzu wurden die einzelnen Modellprognosen mit ihren jeweiligen Prototypaktivierungen gewichtet. Modelle in der lokalen Umgebung des aktiven Modells gehen so automatisch stärker in die Ausgabeberechnung ein als weiter weg liegende Modelle. Zwar verwischt dabei die „scharfe“ Kante der Unstetigkeitsstelle, die globale Modellgüte nimmt jedoch zu. Ein weiterer positiver Aspekt ist, dass dadurch auch weniger Modelle notwendig sind (ca. 10% weniger als im vergleichbaren Versuch aus Abb. 25b).



(a) Zielfunktion (b) 24 Modelle

(c) 50 Modelle (d) Prototyp-Ebene zu (b)

ABBILDUNG 25: In diesem Beispiel wurden einmalig verrauschte Daten der Zielfunktion in (a) präsentiert. Im Vergleich zu (b) wurde bei (c) eine höhere Modellgüte angestrebt, was zu einer feineren Abdeckung mit linearen Modellen führte.

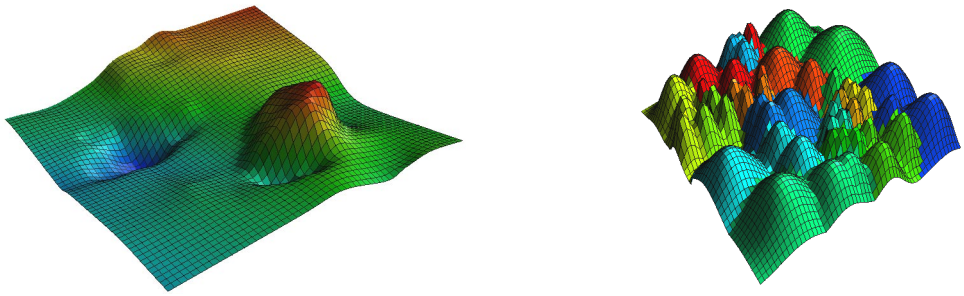
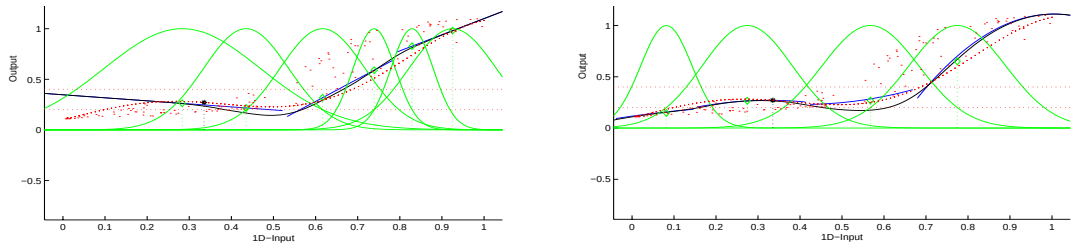


ABBILDUNG 26: Bei diesem Beispiel wurden die Ausgaben aller lokalen Modelle mit ihren Prototypaktivierungen gewichtet und zum Endergebnis aufsummiert.

Zeitvarianz

Um das Verhalten bei zeitvarianten Funktionen zu veranschaulichen, wird das Beispiel aus Abb. 13 verwendet. Hierbei handelt es sich um einen ständig driftenden Prozess. Das lernende System hat die Aufgabe, möglichst den aktuellen Systemzustand zu modellieren. In Abb. 27 sind zwei Ergebnisse dargestellt.

In beiden Fällen ist deutlich zu erkennen, dass der ICE-Algorithmus in der Lage ist, der Drift des Prozesses zu folgen. Bemerkenswert ist, dass jedes Trainingsmuster nur einmal präsentiert wurde. Im Fall von Experiment 27b wurden zusätzlich zu den Prototypen die Quadrate ihrer Offsets zur Adaption der lokalen Modelle verwendet.



(a) Lokale lineare Modelle (b) Lokale nichtlineare Modelle

ABBILDUNG 27: Instationärer Prozess. Identifikation mit lokalen linearen Modellen (a) und nicht linearen Modellen (b).

Vertrauenswert

Wie schon angesprochen ist der ICE-Algorithmus in der Lage, eine Mischung aus Datenverteilung und Approximationsgüte zur Verfügung zu stellen. Diese Eigenschaft lässt sich am besten durch die Präsentation nicht gleichverteilter Daten veranschaulichen. So wurden im Beispiel in Abb. 28 lediglich Daten aus den Intervallen [0:45] und [70:100] selektiert.

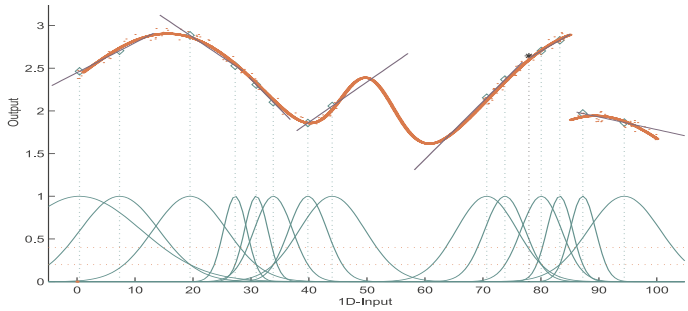
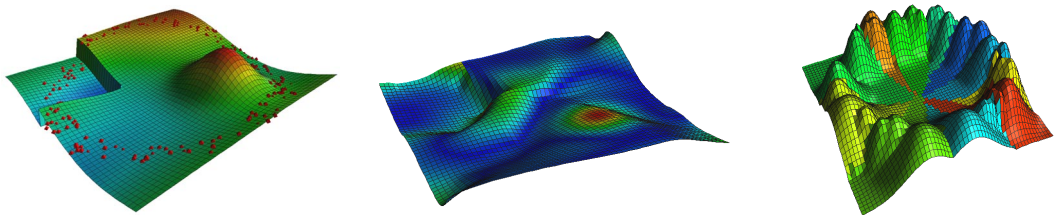


ABBILDUNG 28: In diesem Experiment wurden Daten aus den Intervallen [0:45] und [70:100] zum Training verwendet. Entsprechend sinkt die Prototyp-aktivierung im Intervall [45:70] auf nahezu Null ab (Indiz für unzuverlässige Prognosen).

Wie zu erwarten ist, sinkt im Intervall [45:70] die Aktivierung der Prototypen auf nahezu Null ab. Im Bereich des größten Fehlers liegt die Aktivierung unter Θ^-. Vergleichbare Ergebnisse können auch am Beispiel in Abb. 29 beobachtet werden.



(a) Ziel & Daten (b) Modell (c) Prototypen

ABBILDUNG 29: Ringförmige Trainingsdatenverteilung (vgl. kleine Diamanten in (a)). Das Modell in (b) ist nach dem Fehler eingefärbt. Die Zuverlässigkeit der Prognosen wird durch die Prototypaktivierungen repräsentiert (c).

In diesem Experiment wurde die gleiche Zielfunktion wie in Abb. 25 verwendet. Allerdings wurden hier 200 Trainingsdaten aus einem schmalen Ring (kleine Diamanten in Abb. 29a) selektiert. Entsprechend werden auch die Prototypen in der Cluster-Ebene auf einem Ring er-

zeugt. Wie in Abb. 29b deutlich wird, ist der Fehler im Bereich der Trainingsdaten sehr klein. An anderen Stellen ist dies zum Teil auch der Fall, beruht jedoch auf einem Zufall. Deutlich wird, dass die Prototypaktivierung direkt als Indiz für die Qualität der Netzprognose verwendet werden kann. Kleine Aktivierungen, insbesondere unter der Schwelle von Θ^-, deuten auf Prognosen mit minderer Güte hin.

Hochdimensionale Daten

In weiteren Experimenten mit Eingaberaumdimensionen bis zu 27 Eingängen konnten vergleichbare Ergebnisse wie in den vorangegangenen Beispielen beobachtet werden. Vergleichbar soll heißen, dass schon in der Initiallernphase plausible Ergebnisse zur Verfügung standen. Zum Vergleich wurden unterschiedliche Verfahren getestet. Den kleinsten Fehler auf einer unabhängigen Testmenge erzielte ein MLP-Netz. Allerdings benötigte das Netz ca. 20000 Trainingsdaten und mehrere Epochen für die Trainingsphase. Das bedeutet, dass für die ersten 20000 (von 100000 Datenmustern) keine brauchbaren Netzausgaben zur Verfügung standen. Darüber hinaus mussten zahlreiche Experimente durchgeführt werden, um eine passende Topologie zu finden.

Mit dem ICE-Algorithmus war es dagegen möglich, schon in der Anfangsphase sinnvolle Netzprognosen zur Verfügung zu stellen. Die notwendige Netztopologie wurde jeweils automatisch erzeugt. Bemerkenswert ist, dass obwohl in allen Experimenten die Trainingsdaten dem Netz nur einmal präsentiert wurden, der ICE-Prognosefehler auf den Datenmustern 20000-100000 nur ca. 3-5% höher lag als beim MLP.

Visualisierungswerkzeuge VISP, CUBES

Neben der Entwicklung neuer Verfahren für kontinuierliches Lernen wurden auf der Basis von OpenGL® Visualisierungstools zur interaktiven Analyse zeitvarianter 3D- (VISP) bzw. 4D-Daten (CUBES) entwickelt.

Bei der Analyse unterschiedlicher Verfahren oder großer Datenmengen ist es häufig schwierig, die Flut von Zahlenkolonnen zu überblicken. Dadurch wird das Erkennen von Besonderheiten in den Daten oder spezieller Eigenschaften eines Verfahrens erschwert. Um den Benutzer zu unterstützen, können die Daten bzw. ein Teil der Daten in geeigneter Form visualisiert werden.

Die beiden Varianten des Visualisierungswerkzeugs dienen der Visualisierung von 3D- bzw. 4D-Daten. Das eigentliche Anwendungsgebiet des Werkzeugs ist jedoch die interaktive Analyse des Lernverhaltens unterschiedlicher Algorithmen. Hierzu sind in beiden Versionen Erweiterungen vorhanden, um beispielsweise Trainingsdaten oder aktuelle Fehler mit in die eigentlichen Szenen zu integrieren.

In der aktuellen Version findet die Anbindung an externe Programme wie FAST, Matlab® usw. off-line statt, d.h. das jeweilige Programm schreibt seine zu visualisierenden Daten in eine Datei. Nach Möglichkeit schon im entsprechenden VISP bzw. CUBES-Format. Falls dies nicht möglich oder mit erheblichem Aufwand verbunden ist, sind mehrere leicht erweiterbare Perl-Scripten vorhanden, die die Daten in das benötigte Format umwandeln.

Zusammenfassung und Ausblick

Mit dem ICE-Algorithmus wurde ein inkrementeller Lernalgorithmus zur Identifikation zeitvarianter Prozesse vorgestellt. Das Verfahren wurde speziell für Lernaufgaben im Bereich Kontinuierliches Lernen konzipiert. Für dieses Anwendungsfeld bietet der Ansatz mehrere bemerkenswerte Eigenschaften. So müssen keinerlei Topologieparameter eingestellt werden. Eine adäquate Netztopologie wird in Abhängigkeit der geforderten Prognosegüte automatisch erzeugt. Eine weitere positive Eigenschaft ist, dass bereits in der Initiallernphase sinnvolle Netzprognosen zur Verfügung stehen. Dies wird durch eine starke Interaktion zwischen der Cluster- und der eigentlichen Approximationsebene erreicht. Darüber hinaus ist das Verfahren in der Lage, einen zusätzlichen Ausgabewert als Indiz für die Prognosegüte bereitzustellen.

In der aktuellen Entwicklungsphase werden die radialsymmetrischen Basisfunktionen der einzelnen Prototypen durch achsenparallele asymmetrische Funktionen ersetzt. Ziel ist es, Information detaillierter in den Prototypen speichern und darüber hinaus über die daraus entstehende Darstellung (ähnlich zu Fuzzyregelbasen) die im Netz gespeicherte Information leichter interpretieren zu können. So kann beispielsweise einem Anwender ein einfaches Modell des unbekannten Prozesses in Form einfacher Regeln zur Verfügung gestellt werden.

Ein weiterer Entwicklungsschritt besteht in der Erzeugung und Verwaltung mehrerer Modellebenen. Diese zusätzlichen Modelle stellen eine Art Gedächtnis dar. Ähnlich zum Case-Based-Reasoning könnte dadurch schnell auf vergangene Situationen zugegriffen werden.

4 Analyse iterierter Prozesse

4.1 Iterative Wurzeln

Einführung

Der Bestimmung von Funktionswurzeln liegt das folgende Problem zugrunde: Auf einen bekannten Eingang wird wiederholt eine unbekannte Funktion angewendet. Die Schachtelungstiefe ist dem Anwender bekannt, leider lässt sich aber nur der Ausgangswert nach der letzten Iteration beobachten. Allein aus diesem beobachteteten Zusammenhang sollen Aussagen über die unbekannte Funktion getroffen werden. Das Problem der Funktionswurzelbestimmung kann somit als das inverse Problem zur Komposition von Funktionen angesehen werden.

Da bei der in Abschnitt 5 ausführlich dargestellten Bestimmung der nicht messbaren Zwischenprofile nun genau dieses Problem vorliegt, andererseits jedoch hinter der Ermittlung von Funktionswurzeln ein universelleres Konzept steckt, soll zunächst die reine Theorie näher untersucht werden. Die konkrete Anwendung von Funktionswurzeln auf das Problem der Zwischenprofile wird in Abschnitt 5 erläutert werden.

Definition

Zu einer gegebenen Funktion $F(x): \mathbb{R} \to \mathbb{R}$ ist eine Funktion $f(x)$ mit

$$f(f(x)) \equiv F(x)$$

eine *Funktionswurzel* oder *Iterative Wurzel* von F.

Beispiele: $F(x) = x + 1 \quad \Rightarrow \quad f(x) = x + \frac{1}{2}$

$$F(x) = x^2 \quad \Rightarrow \quad f(x) = |x|^{\sqrt{2}}$$

Iterative Wurzeln höherer Ordnung lassen sich entsprechend definieren: Wenn

$$f^k(x) = f(f(\ldots f(x)\ldots)) \equiv F(x)$$

so ist die Funktion $f = F^{1/k}$ eine *k-te Funktionswurzel von* F.

Beispiel: $F(x) = x + 1 \quad \Rightarrow \quad f(x) = F^{1/k} = x + \frac{1}{k}$

$$F(x) = ax \quad \Rightarrow \quad f(x) = F^{1/k} = \sqrt[k]{a}\, x$$

Diese Definition gilt entsprechend auch für Vektorfunktionen $F(x): \mathbb{R}^n \to \mathbb{R}^n$

Lineare Transformationen F kann man dann als $n \times n$ Matrix schreiben. Daraus die (Funktions)Wurzel zu ziehen, bedeutet, die Matrixgleichung $f \bullet f = F$ oder $f^n = F$ nach f zu lösen.

$$\text{Beispiel: } F\bar{x} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}\bar{x} \quad \Rightarrow \quad F^{1/k}\bar{x} = \begin{bmatrix} \cos\frac{\varphi}{k} & -\sin\frac{\varphi}{k} \\ \sin\frac{\varphi}{k} & \cos\frac{\varphi}{k} \end{bmatrix}\bar{x}$$

Funktionswurzeln stellen das *Inverse Problem zur Iteration* dar: Wenn F die *k-te Iterierte* von f ist, dann ist umgekehrt f die *k-te inverse* oder *fraktionale Iteration* von F

Allgemein kann so der Begriff und die Notation der Iteration von den üblicherweise ganzzahligen Werten erweitert werden: Bekannterweise bedeutet F^{-1} die Inverse, F^0 die Identität und $F^{n\in\aleph}$ die n-te Iterierte der Funktion F. Jetzt kann man $F^{n/k\in\aleph}$ als n-te iterierte der k-ten Wurzel von F definieren, also als *rationale* oder *fraktionale* Iteration.

Existenz

Funktionswurzeln beliebiger Ordnung $f = F^{1/k}$ existieren für alle streng monoton steigenden stetigen reellwertigen Funktionen $F(x):\mathbb{R}\to\mathbb{R}$ [30]. Allerdings ist kein konstruktiver Beweis bekannt, mit dessen Hilfe sich zu einer Funktion deren Wurzel analytisch ableiten ließe.

Im Gegenteil, es ist ausgesprochen schwer, selbst für einfache Funktionen deren Wurzel zu finden. Beispiele hierfür sind z.B. $F(x) = x^2 + 1$ oder $F(x) = e^x$

Falls die Funktion *nicht* streng monoton steigend, stetig und reelwertig ist, gibt es Lösungen oder nicht, ähnlich wie bei „normalen“ Wurzeln und muss von Fall zu Fall untersucht werden.

Von streng monoton fallenden reellwertige Funktionen kann es nur streng monoton fallende Wurzeln ungerader Ordnung geben, komplexwertige haben auch gerade Wurzeln.

Komplexwertige Funktionen der Form $F(x) = ax^2 + bx + c$ haben gar keine Lösungen, reellwertige jedoch mindestens immer dann, wenn sie monoton steigend sind.

Generell gilt die Aussage "... the behavoir of iterative roots may sharply contrast with what one might expect." [30]

Daher ist ein Verfahren, mit dem man auf einfache Weise mit Funktionswurzeln experimentieren kann, auch für Theoretiker sehr interessant. Leider unterstützt keines der bekannten Mathematikpakete, wie Matlab, Mathematica, Maple die analytische oder numerische Berechnung fraktionaler Iterationen. Maple z.B. hat den operator "iterate" deklariert, allerdings nur für ganzzahlige Argumente: F^{-1} bedeutet die Inverse, F^0 die Identität und $F^{n\in\aleph}$ die n-te Iterierte der Funktion F.

Eindeutigkeit

In der Regel sind die Lösungen nicht eindeutig. Zu $F(x) = x$ gibt es die "Wurzeln der Identität" $F^{1/n} = x$, $n \in N$ und $F^{1/k} = -x$, $k \in 2, 4, 6, \ldots$. Aber oft gibt es ganze Funktionsräume von Lösungen. Dies bedeutet oft ein großes Handicap für die Anwendung, da die Frage nach einem Teilprozess nicht eindeutig geklärt werden kann.

In Abschnitt 4.2 (Lineare Funktionen) wird gezeigt, dass es selbst bei so einfachen linearen funktionen $F(x) = x + a$ neben der trivialen Lösung noch beliebig viele oszillierende Lösungen existieren.

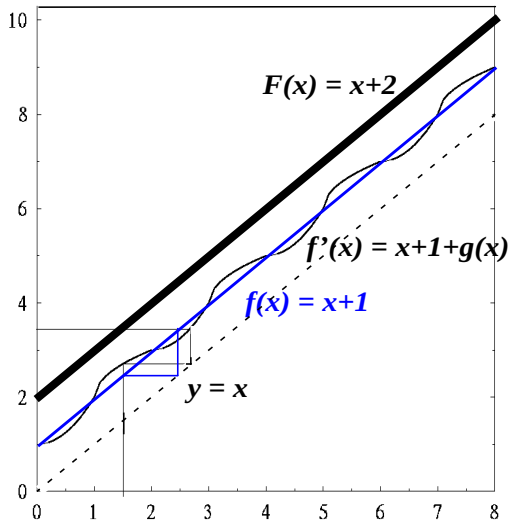


ABBILDUNG 30: Oszillierende Wurzel von y=x+2

Diese oszillierenden Lösungen, die eine eindeutige Bestimmung der inversen Iterierten verhindern, sind jedoch alle um eine "einfachste" Lösung angeordnet, die im Falle eines linearen F ebenfalls linear ist. Eine Möglichkeit, um die Nichteindeutigkeit herumzukommen, wäre stets die Angabe dieser "Hauptlösung". Es gibt natürlich keine Gewähr, dass diese Lösung auch dem tatsächlichen Prozess entspricht, der modelliert werden soll; die Problemstellung ist nicht eindeutig. Unklar ist auch, wie bei anderen als linearen Funktionsverläufen eine solche ausgezeichnete Lösung im Allgemeinen zu definieren ist.

Begrenzte Eingangsdaten

Das Problem verschärft sich nochmals, wenn die Eingangsdaten nur aus einem beschränkten Intervall stammen. Hat eine Transferfunktion nicht überlappende Definitions und Wertebereiche, können beliebige Funktionen als Iterationsfunktionen gewählt werden: Wenn x nur aus dem Intervall $[a]$ stammt, dann kann man eine beliebige Funktion $f'(x)$ wählen, deren Bild von $[a]$, also das Intervall $[b] = f'([a])$ nicht mit $[a]$ überlappt.

Definiere $f''(x) = x + F(x) - f'(x)$ und bilde die Funktion:

$$f(x) = \begin{cases} f'(x) & \text{falls } x \in [a] \\ f''(x) & \text{falls } x \in [b] \\ \text{beliebig sonst,} & \end{cases}$$

so erfüllt dieses f auf dem Intervall $[a]$ auch die Bedingung, dass $f(f(x)) \equiv F(x)$.

4.2 Analytische Betrachtung für lineare Funktionen

Für ein lineares F kann man eine Lösung f noch leicht erraten und aus dieser Lösung weitere konstruieren:

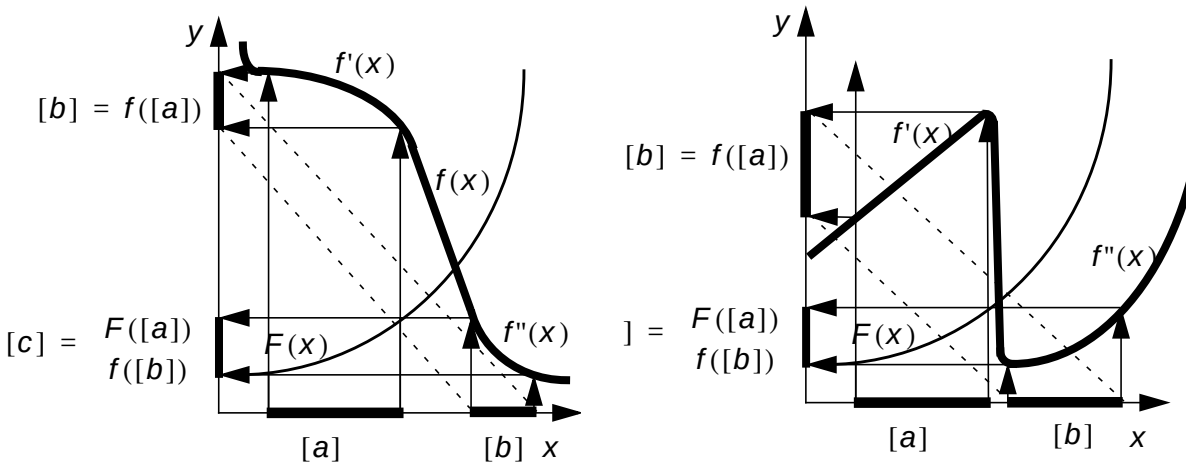


ABBILDUNG 31: Rekonstruktion bei begrenzten Eingangsdaten. Zwei völlig verschiedene f ergeben hintereinandergeschaltet auf dem Intervall $[a]$ das gleiche F.

F(x) = x+c

Ansatz: $f(x) = x + \frac{c}{2} + g(x)$

$$=> f(f(x)) = x + \frac{c}{2} + g(x) + \frac{c}{2} + g\left(x + \frac{c}{2} + g(x)\right) = x + \frac{2c}{2} + g(x) + g\left(x + \frac{c}{2} + g(x)\right) = x + c$$

$$=> g(x) + g\left(x + \frac{c}{2} + g(x)\right) \equiv 0$$

=>Ist x_0 Nullstelle von g, dann auch $x_0 + \frac{kc}{2}$, $k = 1, 2, \ldots$.

=> g ist periodisch mit Periode c.

F(x) = mx

Definiere: $a = \sqrt{m}, F(x) = a^2 x$

Ansatz: $f(x) = ax + g(x)$

$$=> f(f(x)) = a(ax + g(x)) + g(ax + g(x)) = a^2 x + ag(x) + g(ax + g(x)) = a^2 x$$

$$=> ag(x) + g(ax + g(x)) \equiv 0$$

=>Ist x_0 Nullstelle von g, dann auch $a^k x_0$, $k = 1, 2, \ldots$.

=> g ist oszillierend mit sich ändernder Periode.

F(x) = mx+c

Definiere: $a = \sqrt{m}, b = \frac{c}{a+1}, F(x) = a^2 x + b + ab$

Ansatz: $f(x) = ax + b + g(x)$

$$=> f(f(x)) = a(ax + b + g(x)) + b + g(ax + b + g(x)) = a^2 x + ab + ag(x) + b + g(ax + b + g(x))$$
$$= a^2 x + b + ab$$

$\Rightarrow ag(x) + g(ax + b + g(x)) \equiv 0$

$\Rightarrow$Ist x_0 Nullstelle von g, dann auch $ax_0 + b, a(ax_0 + b) + b, \ldots$.

$\Rightarrow g$ ist oszillierend mit wechselnder Periode.

Ist x_a keine Nullstelle von g, dann kann man einen Wert für $g(x_a)$ beliebig vorgeben und daraus eine Kette von Folgewerten berechnen:

$$\Rightarrow F(x_a) = a(ax_a + b) + b = a(ax_a + b + g(x_a)) + b + g(ax_a + b + g(x_a))$$

$$\Rightarrow a^2 x_a + ab + b = a^2 x_a + ab + ag(x_a) + b + g(ax_a + b + g(x_a))$$

$$\Rightarrow g(ax_a + b + g(x_a)) = -ag(x_a)$$

Zwischen zwei Nullstellen (s.o) kann man also g beliebig wählen. Gibt man g für das gesamte Intervall zwischen zwei Nullstellen an, ist g damit komplett bestimmt. Insbesondere kann man g so wählen, dass es z.B. stetig und differenzierbar ist und damit behält auch f diese Eigenschaften. Sogar strenge Monotonie von f ist durch Wahl entsprechend flacher g möglich.

Damit ist der konstruktive Beweis für die Existenz unendlich vieler Lösungen $f(x)$ erbracht, für den Fall, dass $F(x)$ eine lineare Funktion ist.

4.3 Neuronale Berechnung für beliebige Funktionen

Ist die Funktion $F(x)$ als Datensatz von (x, y)-Werten gegeben, lassen sich Multilayer Perzeptrons für die näherungsweise Berechnung der Funktionswurzeln einsetzen. Ausgehend von der Funktionsapproximation für $F(x)$ lässt sich durch eine spezielle Topologie die Wurzel direkt im Inneren des Netzes auslesen (Abb. 32).

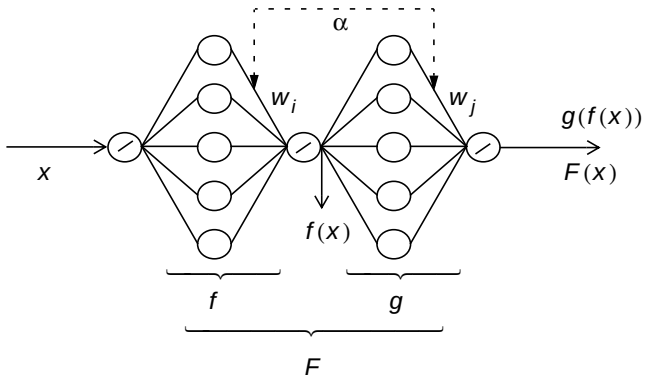


ABBILDUNG 32: Neuronales Netz zur Berechnung von Funktionswurzeln.

Lernmethoden

Netze dieser Topologie sind ein Spezialfall der Multilayer-Perzeptrons. Es gibt keine Unterschiede in der Recall-Phase, nur beim Lernen muss dafür gesorgt werden, dass die korrespondierenden Gewichte der Teilnetze einander entsprechen.

Lernen nur in der letzten Schicht

Die einfachste Methode, eine Funktionswurzelberechnung zu implementieren, besteht darin, nur den hinteren Teil des Netzes (g) aus Abb. 32 zu trainieren und die Gewichte auf die entsprechenden Gewichte des vorderen Teiles zu kopieren. Man kann dies auch mit nur einem einzelnen Netz implementieren: Um die N-te Wurzel zu bestimmen, iteriert man den Eingang N-1 mal durch das Netz, benutzt den resultierenden Wert als Eingang und den gewünschten Ausgang als Target. Wird dieser genau reproduziert, approximiert dieses Netz die N-te Wurzel von F.

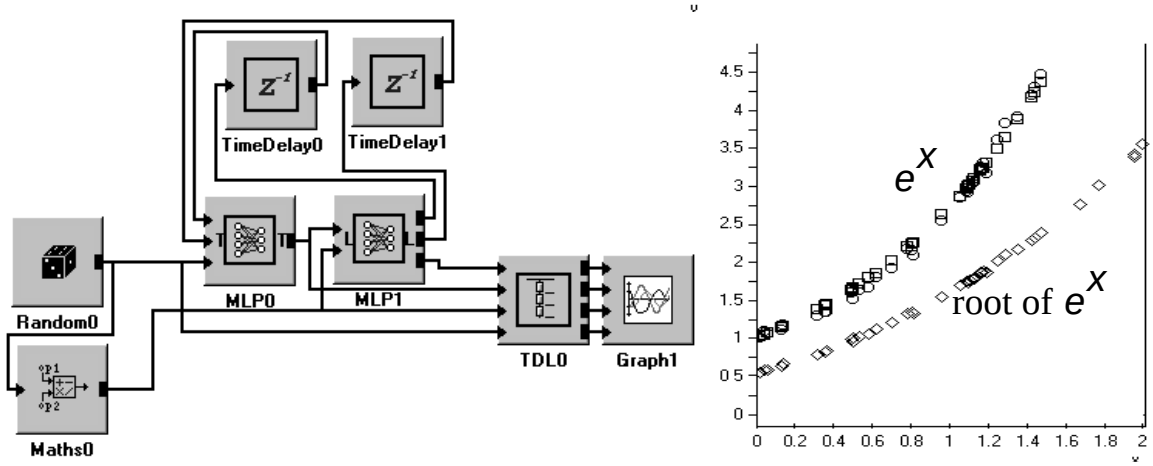


ABBILDUNG 33: Implementation des Verfahrens im SIEMENS Entwicklungssystem ECANSE. In das MATH Objekt kann eine beliebige Funktion eingetragen werden. Nur MLP1 lernt aktiv und kopiert seine Gewichtsmatrix zurück auf MLP0. In der Grafik wird am Beispiel von e^x die Funktion und die Wurzel angezeigt.

Weiches Koppeln der Gewichte

Funktionswurzeln beliebiger Ordnung $f = F^{1/k}$ existieren für alle streng monoton steigenden reellwertigen Funktionen $F(x) : \mathbb{R} \to \mathbb{R}$ [30]. In anderen Fällen gibt es oft keine Lösung. In diesem Fall, kann man je nach Bedarf zwischen Strategien zur Wahl von Kopplungsfaktor α und Lernrate wählen:

$\alpha \ll 1$ und große Lernrate => $g(f(x)) \equiv F(x)$ mit $\|f, g\| \to min$
=> Beste Modellierung von *F(x)*, “Wurzeln” so ähnlich wie möglich

$\alpha \cong 1$ und kleine Lernrate=> $f(f(x)) \equiv G(x)$ mit $\|F, G\| \to min$
=> Approximation von *F(x)* so gut wie möglich durch Iteration von *f(x)*

Zusätzlich zur Wichtungsänderung Δw_k,, die durch das Lernen mit Backpropagation ermittelt wird, erhält man aus der modifizierten Energiefunktion

$$R = \frac{1}{2}\left(\sum_{s=1}^{S} (t_s - y_s)^2 + \sum_{i,k} (w_i - w_k)^2 \right)$$

einen zusätzlichen Term, $\Delta_{share} w_k = \sum_i s_{ik}(w_i - w_k)$, der in jedem Lernschritt zum Gewicht addiert wird: $w_k^{neu} = w_k^{alt} + \Delta w_k + \Delta_{share} w_k$. Dabei gilt:

- $s_{ik} = 0$ bedeutet Standard–Backpropagation-of-Error.
- Mit $s_{ik} = 1$ wird das Gewicht w_k total abhängig von w_i.

- Symmetrisches Annähern der Gewichte durch reziproke Koppelstärken $s_{ik} = s_{ki} \neq 0$.

Ein standard Backpropagation-Netz mit einem Eingang, 5 Hidden Units und einem Ausgang kann die Funktion $F(x) = x^2 + 1$ mit hoher Genauigkeit annähern. Das Hinzufügen von zwei weiteren Layers mit einem und 5 Neuronen ändert das Prinzip und die Genauigkeit kaum. Diese Architektur kann man als das Hintereinanderschalten zweier Funktionen f und g ansehen wie in Abb. 34. Allerdings sind f und g in der Regel nicht identisch. Ferner führen unterschiedliche Initialisierungen der Gewichte auch jedesmal zu anderen Teilfunktionen.

Verbindet man die Gewichte der ersten Hidden Layer mit den entsprechenden der zweiten 5-Neuronen Schicht durch Sharing-Faktoren, nähern sich die Funktionen f und g einander an. Siehe Abb. 35. Die Genauigkeit der Approximation von F geht durch die zusätzliche Randbedingungen etwas zurück. Durch Verstellen des Verhältnisses von Lernrate und Koppelfaktor lassen sich Präferenzen definieren. Bei großen s_{ik} werden f und g ähnlicher, bei kleineren wird F genauer angenähert. Allerdings kann die alte Genauigkeit auch durch Hinzufügen von weiteren Neuronen wieder erreicht werden.

Bei höheren Iterationstiefen tritt ein Problem beim Initiallernen auf. Das Netz braucht sehr lange, um sich überhaupt in Richtung zur Trainingsfunktion zu entwickeln. In der Regel ist die Übertragungsfunktion eines Netzes nach Initialisierung der Gewichte mit kleinen Zufallszahlen eine konstante Funktion. Bei der Architektur mit mehreren Hidden Layers mit nur einem Neuron ist daher der Eingang in die hinteren Schichten praktisch nicht mehr vom Eingang abhängig - das Gradientenabstiegsverfahren bleibt auf einem Plateau stecken. Ein Ausweg ist das Initialisieren der Gewichte mit solchen Werten, dass die Übertragungsfunktion einer Schicht ungefähr die Identitätsfunktion ergibt.

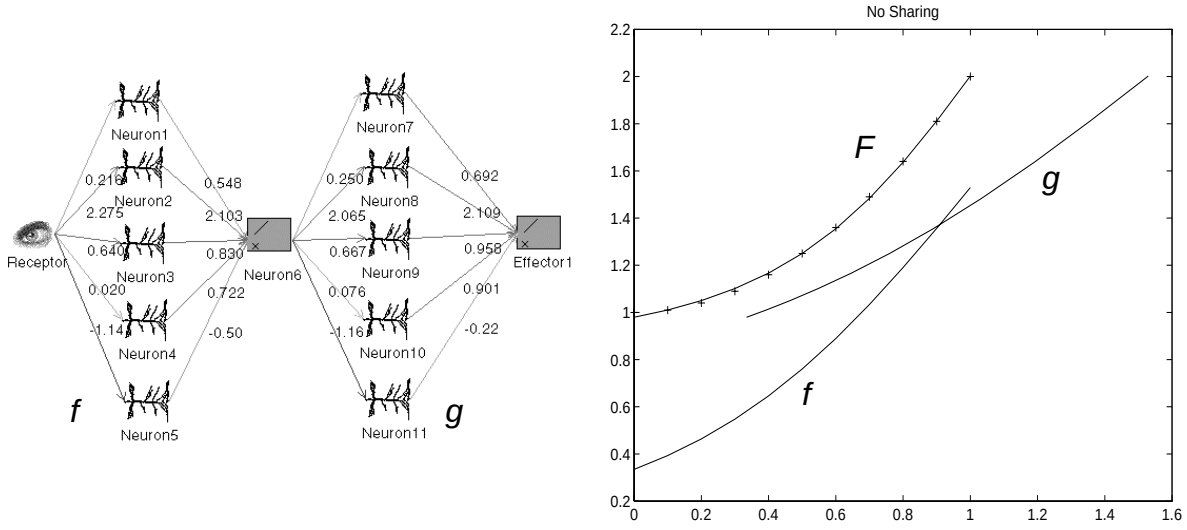


ABBILDUNG 34: Approximation von $(x) = x^2 + 1$ an 11 Trainingswerte mit einem 1-5-1-5-1 Backpropagation Netz. Der RMS Fehler auf dem Trainingsset beträgt 0.008. f und g sind irgendwelche Funktionen mit $F(x) = g(f(x))$, die an Neuron 6 abgegriffen werden.

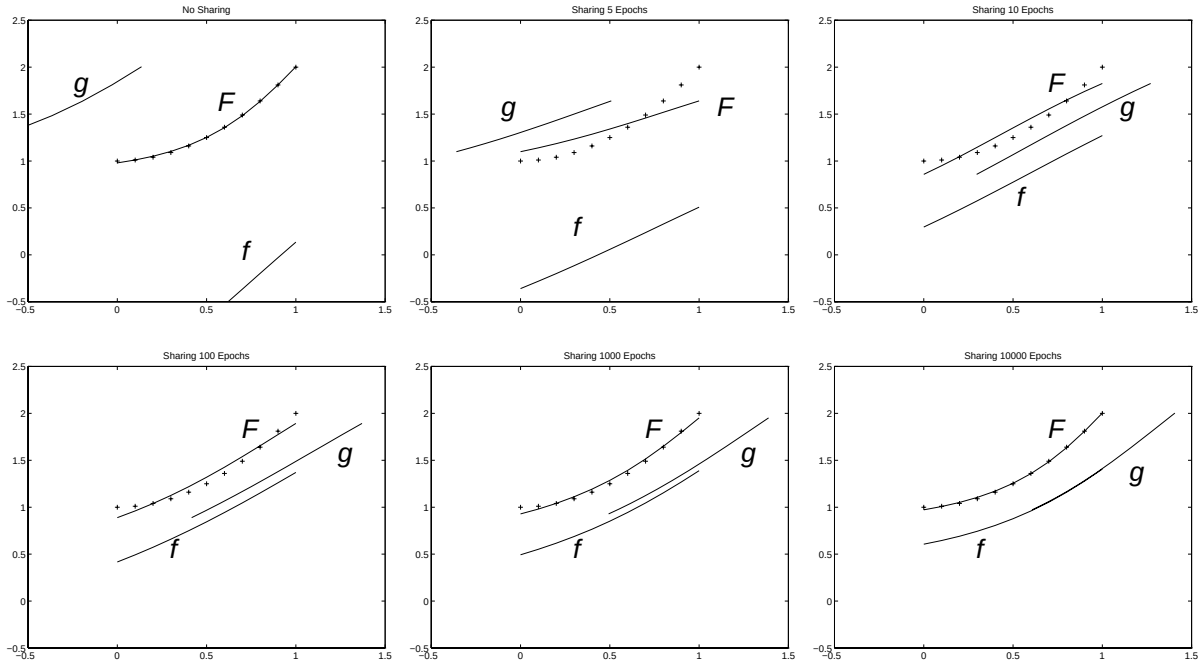


ABBILDUNG 35: Eingeschaltetes Weight-Sharing lässt die Teilübertragungsfunktionen f und g einander annähern. Entwicklung der Gesamt- und Teilfunktionen aus dem Zustand ohne Weight-Sharing nach 5, 10, 100, 1000 und 10000 Epochen. Die Genauigkeit nach 10000 Epochen beträgt 0.011 (RMS) gegenüber 0.008 am Anfang ohne Sharing.

Quasi Newton Methode

Der Algorithmus wird anhand des Falles *einer* Variablen illustriert. Für die Darstellung von f wurde ein einfaches MLP mit Sigmoid-Aktivierungsfunktionen ($\phi(x) = \tanh(x)$ könnte alternativ eingesetzt werden) für die versteckte Schicht und linearem Ausgang verwendet, so dass mit $X_i = f^i(x)$, X_{i+1} rekursiv als

$$X_{i+1} = \sum_{j=1}^{H} c_j \phi(a_j X_i + b_j)$$

geschrieben werden kann.
Falls die k-te Wurzel gesucht ist, lautet die Fehlerfunktion

$$E = \sum_{d=1}^{D} (X_{k,d} - Y_d)^2,$$

wobei $X_{k,d}$ den Netzausgang und Y_d den wahren Ausgangswert für das d-te Beispiel im Datensatz bezeichnen.

Zur Minimierung verwenden wir die Broyden-Fletcher-Goldfarb-Shanno(BFGS)-Prozedur, die in [31] beschrieben wird. Es werden u.a. die partiellen Ableitungen von E bezüglich der Gewichte a_h, b_h und c_h benötigt, beispielsweise

$$\frac{\partial E}{\partial a_h} = \sum_{d=1}^{D} 2(X_{k,d} - Y_d)\frac{\partial X_{k,d}}{\partial a_h}$$

Da die gleichen Gewichte in jedem Teilnetz auftauchen, kann der gewöhnliche Backpropagation-Algorithmus nicht ohne Modifikationen angewendet werden. Da jedoch der Eingang X_0 von den Gewichten unabhängig ist, was zur Folge hat, dass die partiellen Ableitungen

$$\frac{\partial X_0}{\partial a_j} = 0, \frac{\partial X_0}{\partial b_j} = 0 \text{ and } \frac{\partial X_0}{\partial c_j} = 0$$

sämtlich verschwinden, ist es möglich Rekursionsformeln anzugeben (δ_{jh} bezeichnet das Kronecker-Symbol):

$$\frac{\partial X_{i+1}}{\partial a_h} = \sum_{j=1}^{H} c_j \phi'(a_j X_i + b_j)\left(\delta_{jh} X_i + a_j \frac{\partial X_i}{\partial a_h}\right)$$

$$\frac{\partial X_{i+1}}{\partial b_h} = \sum_{j=1}^{H} c_j \phi'(a_j X_i + b_j)\left(\delta_{jh} + a_j \frac{\partial X_i}{\partial b_h}\right)$$

$$\frac{\partial X_{i+1}}{\partial c_h} = \sum_{j=1}^{H} \left(\delta_{jh} \phi(a_j X_i + b_j) + c_j \phi'(a_j X_i + b_j)\left(a_j \frac{\partial}{\partial c_h} X_i\right)\right)$$

Wiederholte Anwendung dieser Formeln erlaubt es nun, die für den Minimierungsalgorithmus benötigten Komponenten

$$\frac{\partial X_k}{\partial a_h}, \frac{\partial X_k}{\partial b_h} \text{ und } \frac{\partial X_k}{\partial c_h}$$

zu berechnen.

Der mehrdimensionale Fall unterscheidet sich hauptsächlich dadurch, dass sich die verwendeten Ableitungen nicht sehr übersichtlich darstellen lassen können, da beispielsweise eine Ableitung $\frac{\partial X_{i+1,t}}{\partial a_h}$ nicht nur von $\frac{\partial X_{i,t}}{\partial a_h}$, sondern auch von allen $\frac{\partial X_{i,s}}{\partial a_h}$ mit $\neq t$ abhängt. Die Strategie zur Berechnung der Ableitungen ändert sich jedoch nicht.

Von den hier vorgestellten Methoden ist diese im Vergleich zu den anderen beiden Methoden bei weitem die schnellste.

4.4 Anwendungsfelder

Die hier vorgestellten Methoden zur Berechnung iterativer Wurzeln sind anwendbar in vielen Bereichen, von "experimenteller Mathematik" bis zur Prozesssteuerung in industriellen Anlagen. Im Folgenden werden einige Beispiele vorgestellt.

Modellierung industrieller Prozesse

Sehr häufig findet man verfahrenstechnische Prozesse, die entweder auf einer wiederholten Anwendung des selben Vorgangs beruhen (Chemie) oder aus einer Hintereinanderschaltung von gleichen Maschinen bestehen (Stahl, Papier). Solche Prozesse anhand von Messdaten des Gesamtprozesses in ihre Teilprozesse aufzuspalten, entspricht häufig der Berechnung iterativer Wurzeln. Im Falle des exemplarischen Anwendungsbeispiels des Projekts, der Profilsteuerung einer Walzstraße, besteht das Problem, aus Mess- und Stellgrößen einer Walzstraße messtechnisch nicht zugängliche Zwischenprofile zu bestimmen. Wie diese Aufgabe auf das Berechnen von iterativen Wurzeln zurückgeführt werden konnte, wird in aller Ausführlichkeit in Abschnitt 5 erläutert werden.

Geometrische Abbildungen

Geometrische Abbildungen sind Vektorfunktionen $\mathbb{R}^n \to \mathbb{R}^n$, die Punkte im Raum auf andere Punkte abbilden. Auch hier kann man die Frage, stellen, ob sich diese Transformation auf mehrere identische Schritte aufteilen lässt. Abb. 36 zeigt ein Beispiel.

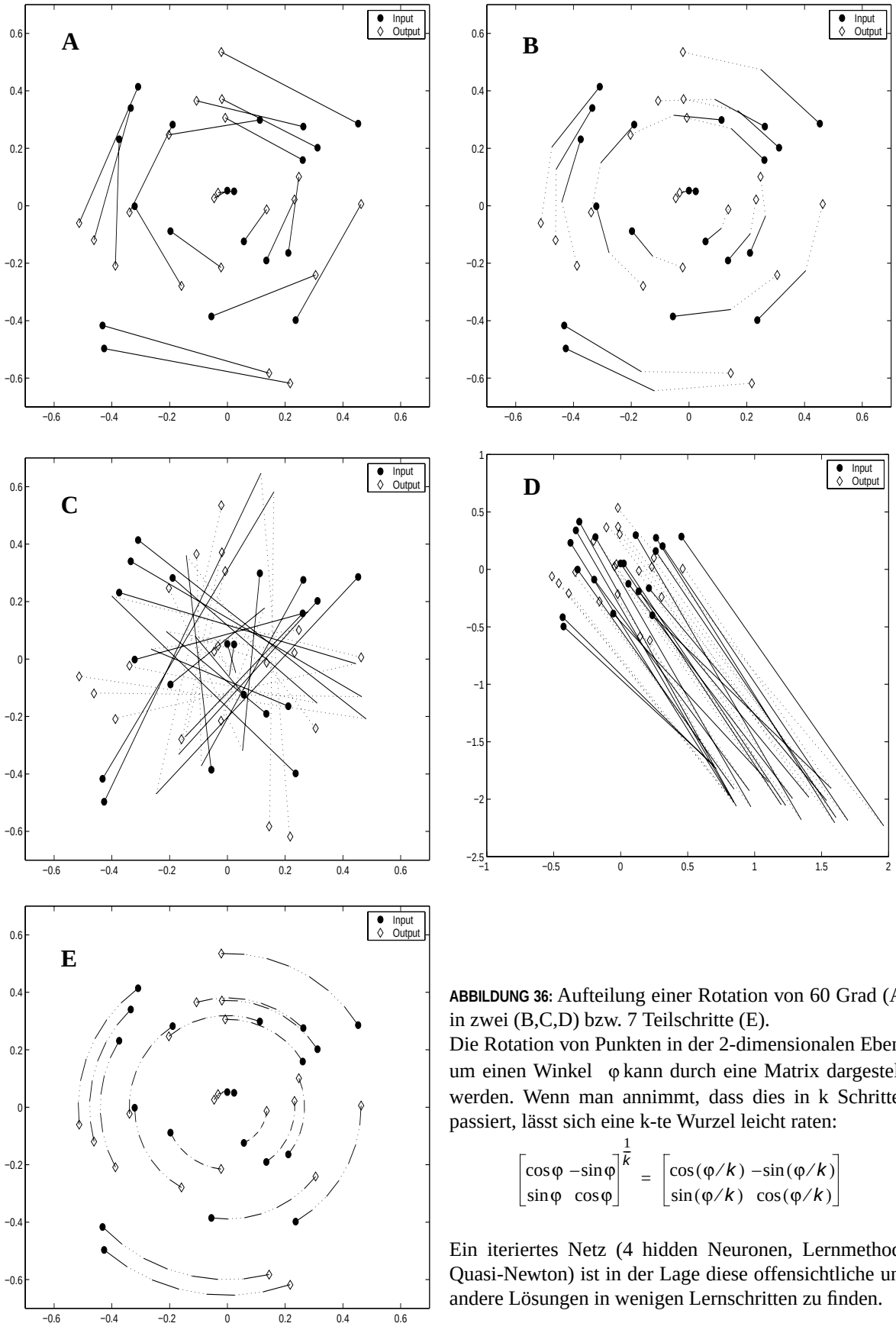


ABBILDUNG 36: Aufteilung einer Rotation von 60 Grad (A) in zwei (B,C,D) bzw. 7 Teilschritte (E).
Die Rotation von Punkten in der 2-dimensionalen Ebene um einen Winkel φ kann durch eine Matrix dargestellt werden. Wenn man annimmt, dass dies in k Schritten passiert, lässt sich eine k-te Wurzel leicht raten:

$$\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}^{\frac{1}{k}} = \begin{bmatrix} \cos(\varphi/k) & -\sin(\varphi/k) \\ \sin(\varphi/k) & \cos(\varphi/k) \end{bmatrix}$$

Ein iteriertes Netz (4 hidden Neuronen, Lernmethode Quasi-Newton) ist in der Lage diese offensichtliche und andere Lösungen in wenigen Lernschritten zu finden.

Chaotische Systeme

Das Iterieren von nichtlinearen Gleichungen ist ein wichtiger Teil der Chaostheorie. Die berühmten chaotischen Systeme nach Feigenbaum und Mandelbrot entstehen durch Iteration der Gleichung $x_{n+1} = \lambda x_n(1-x_n)$ im Reellen bzw. Komplexen [32]. Sind umgekehrt Daten aus einem chaotischen System gegeben, stellt sich die Frage nach dem zugrundeliegenden System. In Abb. 37 wurden Daten durch 4-fache Iteration der logistischen Gleichung erzeugt. Ein scheinbar sehr komplexer Zusammenhang wird durch Invertierung der Iteration auf seinen einfachen, nichtlinearen Ursprung zurückgeführt.

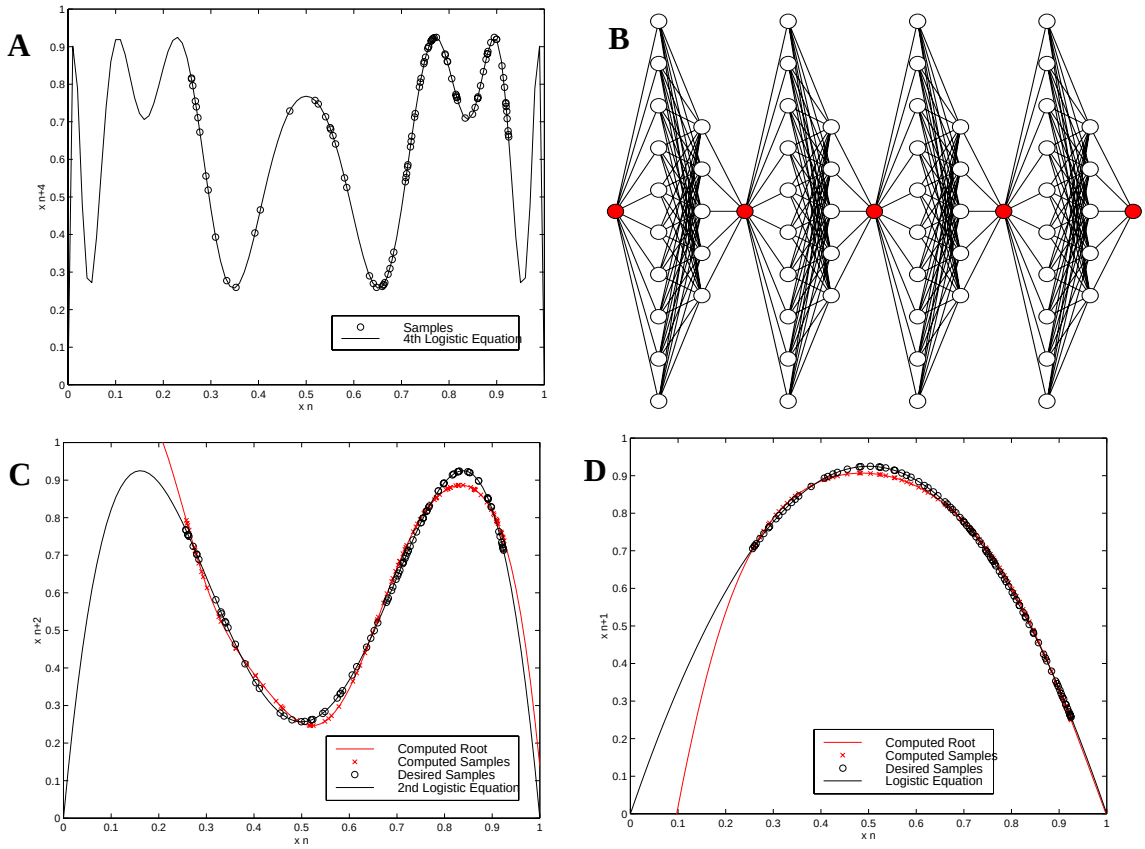


ABBILDUNG 37: (A) die 4-fach iterierten logistischen Gleichung $x_{n+1} = \lambda x_n(1-x_n)$ beim Parameter $\lambda = 3{,}7$ Ein Netz (B) berechnet mit der Weight-Sharing Methode die 2te und 4te Wurzel dieser Funktion (C,D)

5 Industrielle Anwendungen: Profilsteuerung in Walzwerken

5.1 Aufgabenstellung und Datenmaterial

Eine der prominentesten industriellen Anwendungen Neuronaler Netze liegt im Bereich der Walzwerksautomatisierung [33], [34]. Neuronale Netze werden dort als ergänzende Modelle zur Prozessführung eingesetzt. Die zentrale Aufgabe der Prozessführung einer Walzstraße ist die möglichst genaue Ermittlung der Einstellung der Anlage für das jeweils nächste Band *vor* dessen Einlauf in die Straße. Diese Einstellungen werden als Führungsgrößen an die Basisautomatisierung übermittelt, die die Regelung auf der untersten Ebene übernimmt. Die ausgeprägte Nichtlinearität und Zeitvarianz („Tagesform“) des Walzprozesses erfordert ein kontinuierliches Lernen der neuronalen Modelle mit all den Problemen, die in den vorangegangenen Abschnitten diskutiert worden sind.

ABBILDUNG 38: Walzstraße zum Walzen von Stahlblech. Die Maschine besteht aus 7 identischen Walzgerüsten - der Prozess besteht aus 7 hintereinandergeschalteten Teilschritten. Messdaten liegen nur am Ein- und Ausgang vor.

Neben der Breite und der Dicke ist das Profil ein wichtiges Gütekriterium für die Geometrie des Walzgutes nach Verlassen der Fertigstraße. Im einfachsten Fall ist das Profil definiert als der Dickenunterschied zwischen der Mitte und den Rändern eines Bandes. Das Ziel ist, das Profil während des Warmwalzens so klein wie möglich zu halten, da es in späteren Kaltwalzprozessen nicht mehr beeinflusst werden kann. Das Profil des Walzguts nach dem Durchlaufen eines Gerüsts ergibt sich als Funktion der Eigenschaften des Bandes beim Eintritt in das Gerüst und dem

aktuellen Zustand und den Einstellungen des Gerüstes. Abb. 39 zeigt den schematischen Aufbau einer Fertigstraße mit einigen Einflussgrößen auf das Profil. Das dem Stahl aufgeprägte Walzspaltprofil resultiert aus einer mechanischen und thermischen Walzenverformung und spiegelt die Prozessvergangenheit des Walzvorgangs wider. Die thermische Walzenverformung (thermischer Crown) hängt z.B. von den Temperaturen, Breiten und Pausenzeiten einer ganzen Reihe aufeinanderfolgender Bänder ab.

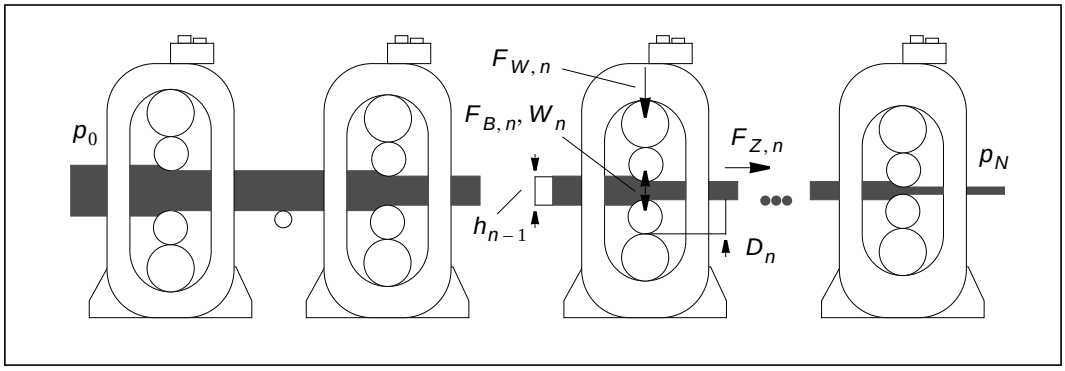


ABBILDUNG 39: Schematische Darstellung einer Fertigstraße Jedes Gerüst n ändert das einlaufende Profil in Abhängigkeit von Parametern wie Banddicke h, Walzkraft F_W, Biegekraft F_B, Zugkraft F_Z, Walzspaltprofil W und Walzendurchmesser D.

Die existierenden mathematischen Modelle, beispielsweise zur Berechnung des Walzspaltprofils, beruhen zwar auf der Modellierung der physikalischen Zusammenhänge, jedoch sind in der Praxis viele idealisierte Voraussetzungen für die Gültigkeit der Modelle nicht erfüllt. Dies führt zu der Verwendung von zusätzlichen, heuristischen Modellen mit einer Vielzahl unbekannter Parameter, deren optimale Einstellung gefunden werden muss. Ein besonderes Problem liegt dabei in der Struktur der Profilmodelle. Während des gesamten Walzprozesses ändert sich das Profil des einlaufenden Stahlbandes von Gerüst zu Gerüst. Erst am Ende der Walzstraße kann es gemessen werden und mit den Vorhersagen des Modells verglichen werden. Das Profil des Bandes zwischen den Gerüsten (Zwischenprofil) ist aus technischen Gründen nicht direkt messbar, jedoch benötigt man für die Stichplanberechnung eine Prognose für den Verlauf des Bandprofils über der gesamten Walzstraße.

Es werden nun häufig Bänder der gleichen Stahlsorte hintereinander gewalzt. Üblicherweise werden die Fehler der Prognosen mit jedem weiteren Band identischen Typs geringer. Große Probleme bereitet bisher jedoch jedes Umstellungsband, beispielsweise kann dies jeweils das erste Band nach einem Wechsel des Stahlsortentyps oder das erste Band nach einer längeren Pause sein. Hier kann es passieren, dass der neue Arbeitspunkt in einem komplett anderen Bereich liegt. Die vom Industriepartner als "Exoten" bezeichneten Bänder fallen *nicht* durch besonders abweichende Messgrößen oder auffallend kleine oder große Eingangsgrößen auf, es sind einfach die Bandtypen, die selten gewalzt werden.

Zur Beurteilung der Güte der hergeleiteten Verfahren wurde vom Industriepartner ein Datensatz A mit 269020 Bändern bei 27 Eingängen zur Verfügung gestellt. Außerdem kam ergänzend ein synthetisch erzeugter Datensatz B mit 100000 Bändern und 11 Eingängen zur Anwendung. Zunächst werden nun Ansätze zur Prognose des Endprofils betrachtet.

5.2 Prognose des Endprofils

5.2.1 Lineare Modelle und ihre Varianten

Lineare Regression

Bevor die Eignung von Neuronalen Netzen und komplizierteren Modellen für die Endprofilprognose untersucht wird, sollen zunächst einfachere Modelle betrachtet werden. Diese zeichnen sich durch ein schnelles Update der Modellgleichungen bei neu eintreffenden Datensätzen aus. Gleichzeitig soll dadurch eine Vergleichsbasis geschaffen werden, um zu überprüfen, in welchem Umfang überhaupt bei komplexeren Konstruktionen eine Verbesserung der Modellprognose zu erwarten ist.

Ein lineares Modell besitzt den Vorteil der leichten Interpretierbarkeit und der kurzen Berechnungszeit. Es bereitet keine Probleme, bestehende Regressionsmodelle bei neu anfallenden Daten anzupassen. Da rekursive Algorithmen existieren, braucht nicht mit jedem neuen Datensatz die rechenzeitintensivere Anpassung mit allen bereits bekannten Datensätzen durchgeführt zu werden. Mit Hilfe dieser rekursiven Algorithmen lassen sich Fenster berücksichtigen, d.h. ältere Datensätze fallen nach einer gewissen Zeit komplett heraus, oder die Datensätze können nach ihrer Aktualität exponentiell nachlassend gewichtetet werden [35] .

Für den Datensatz A ergaben sich für die einfache lineare Regression (mit Adaption, kein Vortraining) folgende Fehlerwerte (RMSE=Root of Mean Squared Error, T=total, U=Umstellungsbänder, N=Nachfolgebänder, E=Exoten, UE=Umstellungs-Exoten):

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
RMSE	0.1269	0.1284	0.1222	0.2116	0.1658

TABELLE 1: RMSE bei der einfachen linearen Regression

Eine visuelle Inspektion der Daten führte dazu, das Quadrat und die Kubik der Variablen V21 als zusätzliche Variablen in das Modell aufzunehmen. Diese Variablen werden ab jetzt immer verwendet. Es ergaben sich dann folgende Fehler:

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
RMSE	0.1190	0.1211	0.1145	0.1885	0.1543

TABELLE 2: RMSE bei linearer Regression mit Hilfsvariablen

Lineare Regression mit Fenstern

Da sich das zugrundeliegende Modell möglicherweise mit der Zeit ändern kann, ist es vielleicht sinnvoll, mit Fenstern einer festen Fensterbreite FB zu arbeiten. Für die Prognose eines Bandes wird immer eine Regression auf den letzten FB Datensätzen durchgeführt. Die Parameterschätzer eines um ein Band nach rechts verschobenen Fensters lassen sich durch eine rekursive Formel angeben. Für die Tabelle 3 wurden verschiedene Fensterbreiten ausprobiert.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
FB=100	0.1037	0.1256	0.0922	0.1387	0.1647
FB=200	0.0985	0.1156	0.0899	0.1334	0.1500
FB=500	0.0973	0.1090	0.0925	0.1337	0.1406
FB=600	0.0973	0.1082	0.0931	0.1351	0.1420
FB=800	0.0982	0.1079	0.0943	0.1370	0.1417
FB=1000	0.0986	0.1074	0.0950	0.1379	0.1407
FB=2000	0.1009	0.1078	0.0971	0.1424	0.1409
FB=5000	0.1052	0.1092	0.1011	0.1651	0.1402
FB=10000	0.1084	0.1116	0.1038	0.1728	0.1438

TABELLE 3: RMSE bei linearer Regression für verschiedene Fensterbreiten

Es zeigt sich, dass der Fehler für die Fensterbreiten 500 und 600 mit 0.0973 deutlich geringer ist als der Wert von 0.1190, der bei einer Regression mit Hilfe *aller* bereits vorhandenen Datensätze auftritt. Es handelt sich aber hier um eine ex-post-Analyse, die optimale Fensterbreite lässt sich erst im nachhinein bestimmen.

Ansätze, die alte Datensätze allmählich, und nicht abrupt, wie bei Verwendung eines Fensters, vergessen, brachten bei den Daten (A) nicht den gewünschten Erfolg.

Die Verwendung eines Fensters hat natürlich zu Folge, dass ältere Datensätze komplett vergessen werden. Es liegt am konkreten Beispiel, ob eventuell das längere Verweilen an einem Arbeitspunkt das gerade verwendete Modell untauglich macht. Speziell kürzere Fensterbreiten sind gegenüber solchen Situationen anfällig.

Betrachtung von Differenzen

Hängt eine Ausgangsgröße linear von den Eingangsgrößen ab, wird aber der Zusammenhang durch einen nichtstationären Fehlerprozess gestört, kann es sinnvoll sein, anstelle der Originalvariablen deren Differenzen zu verwenden. Die Differenz aufeinanderfolgender Ausgangswerte wird durch die Differenz der zugehörigen Eingangsvektoren erklärt. Der nichtstationäre Fehlerprozess wird durch die Differenzenbildung möglicherweise stationär. Eine Prognose für den aktuellen Ausgangswert wird durch Addition des letzten Ausgangswertes mit der Prognose für die Differenz der Ausgänge gebildet.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
RMSE	0.1014	0.1186	0.0922	0.1153	0.1362

TABELLE 4: RMSE bei linearer Regression der Differenzen

Tabelle 4 zeigt, dass die Betrachtung der Differenzen den Fehler im Vergleich zur einfachen linearen Regression (Tabelle 2) senken kann. Auch bei der Regression mit den Differenzen lassen sich Fenster verwenden.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
FB=100	0.1801	0.2366	0.1405	0.1711	0.2203
FB=200	0.1214	0.1520	0.0987	0.1396	0.1736
FB=500	0.1056	0.1262	0.0933	0.1264	0.1523
FB=1000	0.1024	0.1208	0.0924	0.1238	0.1485
FB=1500	0.1015	0.1193	0.0921	0.1211	0.1442
FB=2000	0.1012	0.1187	0.0918	0.1211	0.1443

TABELLE 5: RMSE bei linearer Regression der Differenzen für verschiedene Fensterbreiten

Bei den Differenzen ist es nicht bzw. nur von geringem Vorteil, Fenster für die Regression zu verwenden, zumindest bei den betrachteten Fensterbreiten unterhalb von 2000. Wird das Fenster zu klein, wird das Verfahren anfällig.

Extrapolation der Fehler

Im Abschnitt “Kurzzeitadaption durch Fehlerextrapolation” auf Seite 15 wurde die Methode der Fehlerextrapolation vorgestellt. Um den Begriff des Fehlers noch einmal klar abzugrenzen: Es gibt ein erstes Prognoseverfahren, das von der Fehlerkorrektur nichts weiß und brav seine Voraussagen abgibt. Diese Prognosen werden mit den wahren Werten verglichen und eine Fehlerreihe gebildet. Der extrapolierte Wert wird mit der urspünglichen Voraussage verrechnet. Ein alternativer Ansatz wäre, die Fehler nicht aus der ursprünglichen Prognose zu berechnen, sondern aus der bereits korrigierten Voraussage. Zumindest bei den hier betrachteten Datensätzen war es günstiger, die erste Variante zu verwenden.

Obwohl es sicherlich noch viele andere Verfahren gibt, Zeitreihen vorherzusagen, wurde wieder mit den ARIMA-Modellen gearbeitet. Für die lineare Regression der Differenzen hat sich ein MA(1)-Modell als passend erwiesen. Hier wird angenommen, dass sich der Fehler e_t zum Zeitpunkt *t* durch $e_t = a_t - \theta a_{t-1}$ darstellen lässt, wobei die a_t als unabhängig und identisch normalverteilt angenommen werden. Für die einfache Regression passte ein AR(3)-Modell. Bei

diesem Modell lässt sich der aktuelle Fehler als Linearkombination der letzten drei Fehler plus einer Störgröße darstellen. Das geeignete Modell kann natürlich erst bestimmt werden, wenn einige Daten vorliegen. Die Parameter müssen nicht bekannt sein. Sie können mit den eintreffenden Beobachtungen rekursiv und somit schnell berechnet werden. Dem betrachteten Ansatz liegt die Annahme einer gewissen Stabilität zugrunde: Die Parameter aus den Gleichungen des Fehlermodells dürfen sich ändern, doch sollte zumindest die Grundform des Modells Allgemeingültigkeit haben. Ob sich die Grundform geändert hat, ließe sich aber durch Neuberechnungen der Korrelationsfunktionen feststellen.

Die Berücksichtigung des extrapolierten Fehlers führt zu deutlich verbesserten Ergebnissen, sowohl für die lineare Regression, als auch für die Regression der Differenzen.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
RMSE o.EX	0.1190	0.1211	0.1145	0.1885	0.1543
RMSE m.EX	0.0885	0.1010	0.0852	0.1149	0.1230

TABELLE 6: RMSE ohne und mit Berücksichtigung des extrapolierten Fehlers für die lineare Regression

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
RMSE o.EX	0.1014	0.1186	0.0922	0.1153	0.1362
RMSE m.EX	0.0854	0.1002	0.0798	0.1099	0.1201

TABELLE 7: RMSE ohne und mit Berücksichtigung des extrapolierten Fehlers für die lineare Regression der Differenzen

Fenster für die Fehler

Wird angenommen, dass sich die Art des Fehlerprozesses mit der Zeit ändern kann, bietet es sich an, Fenster für die Fehler zu verwenden, die in die Schätzung der Parameter beim ARIMA-Ansatz eingehen. Beispielsweise werden dann nur die letzten 1000 (oder 100) Fehler zur Schätzung der Korrelationsfunktionen und somit der Parameter verwendet. Kürzere Fensterbreiten führen zu schnelleren Adaptionen der Parameter für das Fehlermodell. Der Einfluss verschiedener Fensterbreiten wurde am Beispiel der linearen Regression mit einem AR(3)-Prozess für die Fehler untersucht. Zusätzlich wurde der Effekt auch für die lineare Regression untersucht, bei der schon die ursprünglichen Daten über ein Fenster der Breite FB=600 gefiltert wurden. Dies war die Fensterbreite, die nach Tabelle 3 zum geringsten RMSE führte.

	Lineare Regression + Fehlerextrapolation mit AR(3)-Prozess					Lineare Regression mit Fenster (FB=600) + Fehlerextrapolation mit AR(3)-Prozess				
FB Fehler	**T**	**U**	**N**	**E**	**UE**	**T**	**U**	**N**	**E**	**UE**
10	0.1021	0.1096	0.1002	0.1365	0.1349	0.0960	0.1076	0.0911	0.1310	0.1433
50	0.0933	0.1029	0.0898	0.1230	0.1274	0.0918	0.1047	0.0859	0.1248	0.1411
100	0.0906	0.1014	0.0865	0.1169	0.1252	0.0901	0.1037	0.0839	0.1226	0.1397
500	0.0885	0.1007	0.0851	0.1142	0.1236	0.0884	0.1027	0.0824	0.1201	0.1381
1000	0.0882	0.1006	0.0849	0.1135	0.1235	0.0882	0.1028	0.0821	0.1199	0.1380
2000	0.0882	0.1006	0.0849	0.1129	0.1233	0.0881	0.1027	0.0820	0.1196	0.1376
5000	0.0882	0.1007	0.0851	0.1133	0.1230	0.0880	0.1027	0.0820	0.1193	0.1374
10000	0.0884	0.1008	0.0854	0.1138	0.1228	0.0880	0.1027	0.0821	0.1192	0.1373
∞	0.0885	0.1010	0.0852	0.1149	0.1230	0.0880	0.1028	0.0819	0.1188	0.1372

TABELLE 8: Vergleich des RMSE für verschiedene Fensterbreiten der Fehler

Zwei Dinge sind aus Tabelle 8 ablesbar: Zum einen ist der deutliche Vorteil, den die lineare Regression mit Fenster gegenüber der normalen Regression besaß, durch die Extrapolation der Fehler mittels des AR(3)-Prozesses nahezu verschwunden (siehe Tabellen 2 und 3). Weiterhin lohnt es sich nicht, für die Fehler ein Fenster zu verwenden; der Gewinn an Prognosegüte ist im Fall der normalen Regression für Fensterbreiten zwischen 1000 und 2000 nur sehr gering, im Fall der linearen Regression mit Fenster zu vernachlässigen.

Da außerdem die normale Regression mit Extrapolation der Fehler für die interessanteren Fälle der Umstellungsbänder und der Exoten die geringeren Fehlerraten aufweist, brauchen Fenster sowohl bei den ursprünglichen Daten als auch bei den Fehlern nicht mehr berücksichtigt zu werden. Dies deutet daraufhin, dass sich eher der Offset des funktionalen Zusammenhangs verändert als die Form der Funktion.

m-Schritt-Prognosen der Differenzen

Es ist natürlich möglich, nicht nur einen Schritt, sondern mehrere Schritte in die Zukunft vorauszusagen. Dies wurde für die Differenzen durchgeführt. Wurde bisher aus dem Differenzvektor des aktuellen Eingangs zum folgenden Eingang in Verbindung mit dem aktuellen Ausgangswert der folgende Ausgangswert vorausgesagt, so hätte die Prognose auch schon ein Band vorher erstellt werden können, mit dem Differenzvektor des vorherigen Eingangsvektors zum nächsten Eingangsvektor, in Verbindung mit dem vorherigen Ausgangswert. Die zu diesen Vorgehensweisen gehörigen Prognosen werden nun 1- bzw. 2 Schritt-Prognosen genannt (an dieser Stelle immer in Verbindung mit dem Ansatz, der die Differenzen verwendet). Allgemein können m-Schritt-Prognosen betrachtet werden. Für Werte von m zwischen 1 und 5 wurden die RMSE berechnet (siehe Tabelle 9).

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
1-Schritt	0.1014	0.1186	0.0922	0.1153	0.1362
2-Schritt	0.1051	0.1187	0.1119	0.1295	0.1348
3-Schritt	0.1094	0.1198	0.1123	0.1381	0.1373
4-Schritt	0.1133	0.1244	0.1129	0.1485	0.1432
5-Schritt	0.1163	0.1255	0.1168	0.1547	0.1405

TABELLE 9: RMSE für verschiedene Prognosehorizonte bei Verwendung von Differenzen

Es ist zu erkennen, dass die Prognosegüte mit zunehmenden Prognosezeitraum abnimmt. Speziell der RMSE des ersten Nachfolgebandes (N) ist deutlich geringer, falls die 1-Schritt-Prognose verwendet wird. Dies ist nicht besonders verwunderlich, da der vorhergehende Eingangsvektor auf jeden Fall zu einem Band der gleichen Stahlsorte gehört, die Differenz der Eingangsvektoren somit klein ist und deshalb die Prognose nur wenig vom Ausgangswert des vorhergehenden Bandes abweichen wird.

Ein einfaches „Mixture of Experts“

Liegen mehrere Prognoseverfahren vor, so kann eine einfache Linearkombination der Einzelprognosen die Prognosegüte erheblich verbessern. Dies soll am Beispiel der Prognosen, die über die Differenzen gebildet wurden, demonstriert werden. Zu diesem Zweck wird der tatsächliche Ausgangswert über eine lineare Regression durch die Einzelprognosen erklärt. Als Startwerte im konkreten Beispiel der Differenzen erhält jede Einzelprognose das Gewicht 0.2. Es können wieder die Vorteile der linearen Regression genutzt werden: die rekursiven Formeln erlauben eine schnelle Berechnung der Koeffizienten, einer schnellen adaptiven Anpassung steht somit nichts im Wege.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
Mixture	0.0851	0.0992	0.0802	0.1099	0.1200
1-Schritt-Prognose + MA(1)-Extrapolation	0.0854	0.1002	0.0798	0.1099	0.1201

TABELLE 10: RMSE für Mixture aus 1-5-Schritt-Prognosen und für die 1-Schritt-Prognose mit Fehlerextrapolation

Der RMSE des Mixture-Ansatzes ist mit 0.0851deutlich kleiner als der mit 0.1014 kleinste Fehler der 5 Einzelverfahren. Tabelle 10 zeigt weiterhin, dass die Ergebnisse des Mixture-Ansatzes sehr ähnlich sind zur 1-Schritt-Prognose mit anschließender Extrapolation des Fehlers. Die Koeffizienten, die mit $a_1 = a_2 = a_3 = a_4 = a_5 = \frac{1}{5}$ initialisiert worden sind, besitzen zum Schluss die Werte $a_1 = 0.35$, $a_2 = 0.25$, $a_3 = 0.17$, $a_4 = 0.12$ und $a_5 = 0.11$. Die 1-Schritt-Prognose bekommt somit im Laufe der Zeit die höchste Gewichtung. Die Abb. 40 zeigt den Verlauf für die

Koeffizienten aus der Regression. Nach einer ersten unruhigen Einschwingphase bleiben die Koeffizientenwerte recht stabil, abgesehen von zwei Erschütterungen, die die Parameter auf ein neues Niveau setzen.

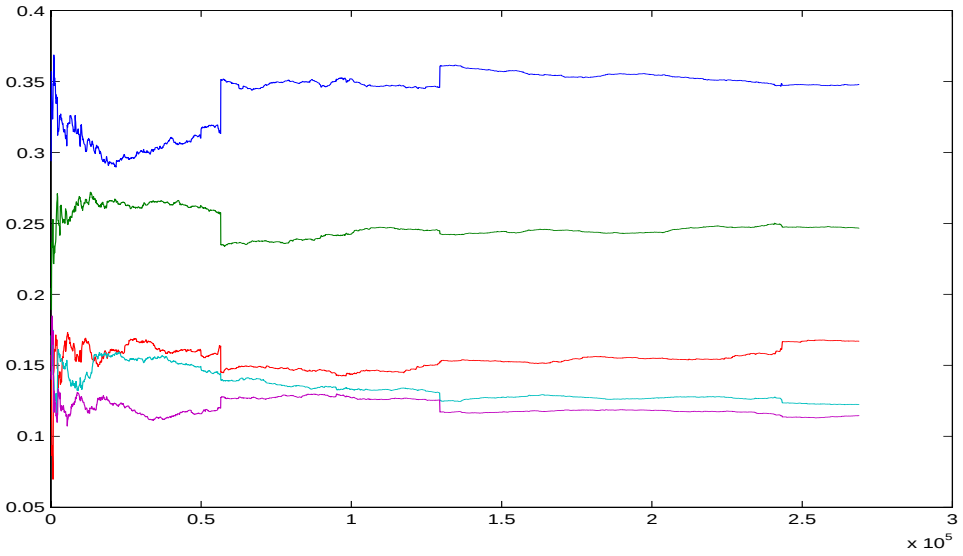


ABBILDUNG 40: Zeitliche Änderung der Koeffizienten

Ausreißerbehandlung

Für den Datensatz ist bekannt, dass einige Eingangswerte nicht bzw. nicht korrekt aufgenommen wurden. Diese sind einfach auf den Wert null gesetzt. Obwohl es über 14.000 Fälle sind, sinkt der Fehler nicht, wenn diese Fälle vom Update der Regressionskoeffizienten ausgeschlossen werden. Die Tabelle 11 gibt den durchschnittlichen Fehler für alle Fälle an, also inklusive der Ausreißer. Aber seltsamerweise hat der Fehler auf den Daten ohne Ausreißer immer noch einen Wert von 0.1192. Es ist hier deshalb besser, alle Werte zu berücksichtigen.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
Lernen mit Ausreißern	0.1190	0.1211	0.1145	0.1885	0.1543
Lernen ohne Ausreißer	0.1195	0.1211	0.1146	0.1889	0.1543

TABELLE 11: RMSE mit und ohne Ausreißer

Die Abbildung der Entwicklung der Koeffizienten im „Mixture of Experts“-Modell zeigte, dass mindestens zwei Regionen existieren, bei denen sich die optimale Linearkombination abrupt ändert. Eine nähere Inspektion der Daten deckt auf, dass in diesen Bereichen der Ausgang auf

sehr hohe Werte über 5 steigt, während im Normalfall Werte kleiner als eins üblich sind. Werden die 11442 Exoten, wozu auch diese Fälle gehören, vom Lernen ausgeschlossen, so ergibt sich das Bild aus Tabelle 12:

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
Lernen mit Exoten	0.1190	0.1211	0.1145	0.1885	0.1543
Lernen ohne Exoten	0.1202	0.1216	0.1151	0.2102	0.1641

TABELLE 12: RMSE mit und ohne Exoten

Wird nur mit Nicht-Exoten gelernt und auch nur der Fehler für die Nicht-Exoten berechnet, so ist der Fehler mit 0.1145 nur wenig geringer als der Wert von 0.1150 für den Fall, dass auch Exoten beim Lernen berücksichtigt werden. Die Berücksichtigung der Exoten verbessert natürlicherweise die Prognosen für die Exoten, verschlechtert andererseits nur geringfügig die Ergebnisse für Nicht-Exoten.

Einfluss der Fehlerextrapolation und der Lernintensität

An dieser Stelle wird zunächst geschaut, welchen Einfluss das ständige Weiterlernen mit allen neuen Daten hat. Zum Vergleich wird eine Lineare Regression durchgeführt, die adaptiv angepasst wird, bis 20000 Datensätze erreicht sind. Dann wird das Modell eingefroren und die Prognosen für die folgenden Bänder mit diesem festen Modell erstellt.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
LR alle	0.1190	0.1211	0.1145	0.1885	0.1543
LR bis 20000	0.1366	0.1350	0.1311	0.2404	0.1788
LR alle + AR(3)	0.0885	0.1010	0.0852	0.1149	0.1230
LR bis 20000 + AR(3)	0.0891	0.1027	0.0847	0.1196	0.1286

TABELLE 13: RMSE für komplettes und verkürztes Lernen

Auf den ersten Blick scheint es so, als ob das verkürzte Lernen mit nur 20000 Datensätzen zu einer merklich schlechteren Prognose führt (RMSE=0.1366 im Vergleich zu 0.1190). Werden jedoch die Fehler extrapoliert, etwa mittels eines AR(3)-Modells, so ist der Fehler mit nunmehr 0.0891 im Vergleich zu 0.0885 nur noch geringfügig erhöht.

Das kann mindestens zwei Gründe haben: Einerseits könnte es sein, dass die Form des funktionalen Zusammenhangs linear ist und sich nur wenig ändert. Nur wird der Zusammenhang durch einen Fehlerprozess gestört, der sich durch ein AR(3)-Modell gut beschreiben lässt. Nach 20000 Datensätzen haben sich die Schätzer der Koeffizienten des linearen Modells halbwegs stabilisiert, und nur der schwankende Offset ist einer exakten Modellbestimmung im Wege. Der Offset lässt sich aber gut durch das AR(3)-Modell vorhersagen. Diese Annahmen haben eigentlich zur Idee des Ansatzes der Fehlerextrapolation geführt.

Ein zweiter Grund kann aber auch sein, dass die Annahme des linearen Modells nicht zulässig ist. In Wirklichkeit liegt ein nichtlinearer Zusammenhang vor. Befindet sich nun der Arbeitspunkt an einer Stelle, an der das lineare Modell schlecht passt, wird jedoch eine größere Anzahl von Bändern des gleichen Typs gefertigt und somit der Arbeitspunkt nicht verlassen, so wird wiederholt mittels des linearen Modells der gleiche falsche Ausgangswert vorausgesagt. Vernachlässigt man sonstige Störungen, so wird sich wiederholt der gleiche Fehler einstellen. Mit wachsender Wiederholung erkennt das AR(3)-Modell die Struktur der Fehler (hier in diesem Beispiel eine konstante Folge) und kann den nächsten Fehler immer besser voraussagen.

Liegen deshalb häufiger hohe Laufzeiten einer Bandsorte vor, so wird der Einfluss eines fehlspezifizierten Modells zur originären Prognose geringer.

Die beiden genannten Ursachen können sich auch noch überlappen und gegenseitig beeinflussen.

Ein Experiment

Um den Einfluss der Fehlerextrapolation noch einmal von einer anderen Seite zu beleuchten, wird vom einfachsten Modell, einer Konstanten, ausgegangen und der Fehler extrapoliert. Die Konstante wird einfach als Mittelwert aller Ausgangswerte angenommen.

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
Konstante	0.2063	0.1923	0.1995	0.4281	0.3263
Konstante + AR(3)	0.1172	0.1431	0.1042	0.1612	0.1860
Lineare Regression	0.1190	0.1211	0.1145	0.1885	0.1543

TABELLE 14: RMSE für eine Konstante

Wie die Tabelle 14 zeigt, führt allein die Verwendung der Fehlerextrapolation für ein Modell, das aus einer Konstanten besteht, zu einer sehr deutlichen Verringerung des Fehlers. Der begangene Fehler ist mit 0.1172 sogar geringer als der Fehler 0.1190 bei der linearen Regression ohne Fehlerkorrektur. Die Verbesserungen durch das AR(3)-Modell sind also zum Teil auf das Verharren an einem Arbeitspunkt für längere Zeit zurückzuführen. Dafür spricht, dass der RMSE mit der Bandnummer sinkt (siehe Abbildung Abb. 41, zweite Kurve von unten). Nur bei Umstellungsbändern ist der Fehler der einfachen linearen Regression (dritte Kurve von unten) geringer, für jedes folgende Band erhält man mit der Fehlerextrapolation auf der Basis einer konstanten Voraussage den geringeren Fehler. Obwohl die Verwendung einer Konstanten nur wenig mit einem physikalischen Modell zu tun hat, ist die Prognosegüte zumindest für Folgebänder brauchbar.

Es kann aber trotzdem sein, dass das AR(3)-Modell weiterhin zufällige Verschiebungen des gesamten funktionalen Zusammenhangs abfängt.

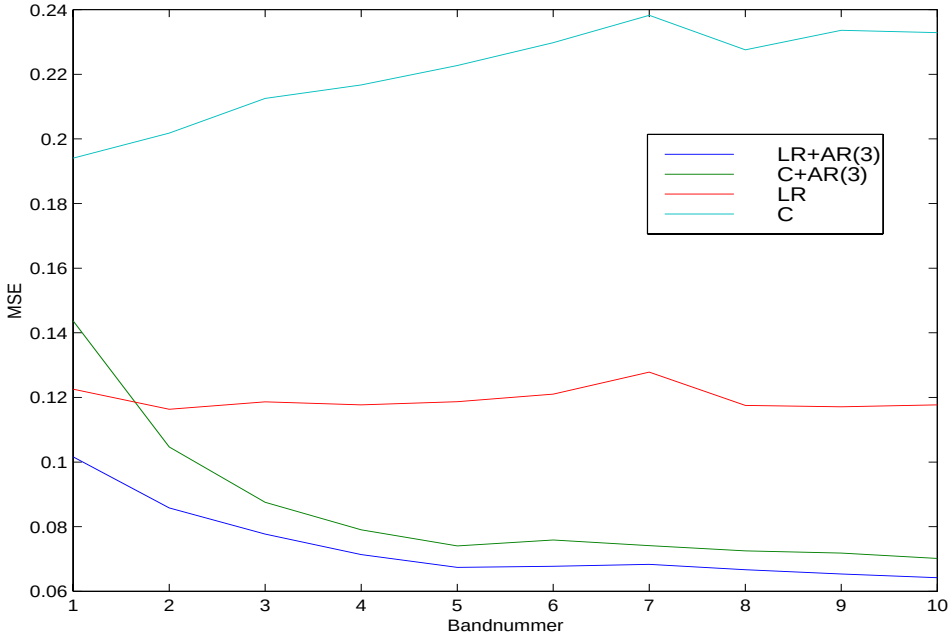


ABBILDUNG 41: RMSE-Fehler in Abhängigkeit von der Bandnummer, mit und ohne Fehlerkorrektur

Kombination von mehreren Modellen zu einer verbesserten Prognose

Das Beispiel auf S. 66 zeigte, dass mit einer einfachen Regression mittels einer Linearkombination aus den erwähnten 5 Prognosen eine verbesserte Prognose zu erzielen sind. Es wäre aber auch denkbar, die Güte der einzelnen Prognosen im Zeitverlauf zu beurteilen und jeweils das zu einem bestimmten Zeitpunkt beste Modell auszuwählen. Der einfachste Ansatz besteht darin, einfach das Modell für die aktuelle Prognose auszuwählen, das den letzten Wert am besten vorausgesagt hatte. Das entstehende Modell „Bestlast“ hat jedoch keine überzeugende Performance, wie die folgende Tabelle zeigt:

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
1-Schritt	0.1014	0.1186	0.0922	0.1153	0.1362
2-Schritt	0.1051	0.1187	0.1119	0.1295	0.1348
3-Schritt	0.1094	0.1198	0.1123	0.1381	0.1373
4-Schritt	0.1133	0.1244	0.1129	0.1485	0.1432
5-Schritt	0.1163	0.1255	0.1168	0.1547	0.1405
Mixture	0.0851	0.0992	0.0802	0.1099	0.1200

TABELLE 15: RMSE-Fehler für 1-5-Schritt-Prognosen, deren Mixture über eine Lineare Regression und für das Modell „Bestlast“

	T	U	N	E	UE
Mittelwert	0.0865	0.0994	0.0840	0.1159	0.1207
Bestlast	0.1078	0.1215	0.1106	0.1324	0.1398

TABELLE 15: RMSE-Fehler für 1-5-Schritt-Prognosen, deren Mixture über eine Lineare Regression und für das Modell „Bestlast“

Zusätzlich ist noch der Fehler für das Modell angegeben, das aus einfacher Mittelung der 1 bis 5-Schritt-Prognosen entsteht. Selbst dieser ist nur etwas größer als der Fehler des Mixture-Modells und im Vergleich zu „Bestlast“ deutlich kleiner. Dass einzelne Modelle nicht für längere Zeit anderen Modellen überlegen sind, zeigt die Tabelle der Übergangshäufigkeiten. Die Zeile gibt an, welches der 5 Modelle den letzten Wert am besten vorhergesagt hat und die Spalte gibt an, welches Modell den aktuellen Wert am besten vorhersagt:

t-1 \ t	1-Schritt	2-Schritt	3-Schritt	4-Schritt	5-Schritt
1-Schritt	16167	13467	12325	11281	10917
2-Schritt	12726	12763	10497	10932	10127
3-Schritt	11027	11027	10803	9103	9714
4-Schritt	10599	10555	9643	9465	8578
5-Schritt	13638	9233	8406	8059	7967

TABELLE 16:

Optimal wäre der Ansatz, wenn die Tabelle nahezu Diagonalform aufwiese. Dies würde bedeuten, dass Modelle jeweils längere Zeit die besten Modelle bleiben. Davon ist man jedoch in diesem Falle weit entfernt. Da starkes Rauschen hier eine große Rolle spielen könnte, wäre es eine Alternative, das für die aktuelle Prognose verantwortliche Modell über die Performance über einen längeren Zeitraum auszuwählen. Das Modell, das im Schnitt der letzten m Prognosen den geringsten absoluten Fehler erzielt hat, könnte dann für die aktuelle Prognose ausgewählt werden.

5.2.2 Neuronale Netze

Adaptives Lernen mit Vortraining

Mit dem bisher betriebenen Aufwand lässt sich die Eignung von Neuronalen Netzen und ihr etwaiger Vorteil gegenüber einfachen linearen Modellen und ihren Erweiterungen besser beurteilen. Mit den ersten 20.000 Trainingsdaten des Datensatzes A wurden Netze verschiedener Architektur vortrainiert. Anschließend wurde online mit verschiedenen Fensterbreiten mit den neu eintreffenden Datensätzen nachtrainiert. Die Anzahl der Epochen pro Fenster wurde ebenfalls variiert, ebenso wie die Lernrate des Backpropagation-Algorithmus. Alternativ wurde der Resilient-Backpropagation-Algorithmus getestet, aber dieser schien für die betrachteten Fensterbreiten ungeeignet. Das beste Ergebnis wurde bei einer Fensterbreite von 10, einer Lernrate von 0.0005 und einer Epochenanzahl von 6 erzielt. Die Fehler der Netzprognose wurden mit

Hilfe eines AR(2)-Modells und zweier AR(1)-Modelle extrapoliert. Das erste AR(2)-Modell wurde nur für Bänder mit einer Laufzeit von 3 und mehr, die anderen beiden Modelle jeweils für Bänder mit einer Laufzeit von 2 bzw. 1 angewendet. Es ergab sich die folgende Tabelle:

	T	U	N	E	UE
Anzahl	269020	125744	50516	11442	5665
Neuronales Netz	0.0820	0.0913	0.0807	0.1121	0.1142
Neuronales Netz + AR	0.0778	0.0894	0.0738	0.1005	0.1127

TABELLE 17: RMSE-Ergebnisse des besten Netzes

Zu berücksichtigen ist, dass diese Werte auch die 20.000 Muster enthalten, mit denen vortrainiert wurde. Die Fehler ohne diese 20.000 Muster liegen aber nur geringfügig höher:

	T	U	N	E	UE
Anzahl	249020	117007	46742	10683	5341
Neuronales Netz	0.0824	0.0918	0.0811	0.1138	0.1151
Neuronales Netz + AR	0.0782	0.0899	0.0740	0.1019	0.1137

TABELLE 18: RMSE-Ergebnisse des besten Netzes, ohne Trainingsdaten

Die Ergebnisse sind natürlich etwas zu optimistisch, da ja das Optimum aus einer Vielzahl von Einstellungen ausgewählt wurde.

Einfluss des Parameters Epochenzahl auf die Güte der Prognose

Der Parameter Epochenanzahl, der beschreibt, wie häufig ein neuer Datensatz bzw. eine Menge von Datensätzen zum Nachtrainieren verwendet wird, hat sicherlich einen Einfluss auf die Prognosegüte. Alternative Ansätze, wie etwa das Trainieren bis zur Unterschreitung einer Fehlergrenze auf den Datensätzen, könnten ebenfalls betrachtet werden. Abb. 42 zeigt den Verlauf des RMSE für verschiedene Epochenanzahlen. Gleichzeitig ist der Fehler eingetragen, der sich nach einer modifizierten AR(2)-Fehlerextrapolation (unterschiedliche Parameter für Umstellungsbänder und Nachfolgebänder) ergibt. Es wurde der Backpropagation-Algorithmus verwendet, die Fensterbreite betrug 10 und die Lernrate 0.0005. Zu sehen ist, dass die oben gelegene Fehlerkurve des reinen Lernverfahrens ohne Fehlerextrapolation recht empfindlich auf die richtige Wahl der Epochenanzahl reagiert. Ein zu vorsichtiges Nachtrainieren hat im Vergleich zum AR-Ansatz mit Fehlerextrapolation schon sichtbare negative Folgen. Außerdem fällt auf, dass für die betrachteten Werte der Epochen der RSME-Fehler des Ansatzes mit Fehlerextrapolation immer unter den Werten des reinen Lernverfahrens liegt. Selbst der schlechteste Wert *mit* Fehlerextrapolation liegt noch unterhalb des besten Wertes *ohne* Fehlerextrapolation. Da es immer eine optimale Epochenanzahl gibt, die natürlich vorher nicht bekannt ist, sollte man den Ansatz mit Fehlerextrapolation vorziehen. Der Fehler wird geringer und die Sensitivität bezüglich der richtigen Epochenanzahl ist für die Praxis ausreichend klein.

Es haben sich ähnliche Ergebnisse bezüglich der Variation anderer Parameter ergeben. Ein Veränderung der Lernrate hat längst nicht so dramatische Folgen, wenn die Fehler extrapoliert werden. Eine zu große Lernrate mit resultierendem chaotischen Verhalten kann andererseits auch nicht durch den Einsatz der ARIMA-Modelle abgefangen werden.

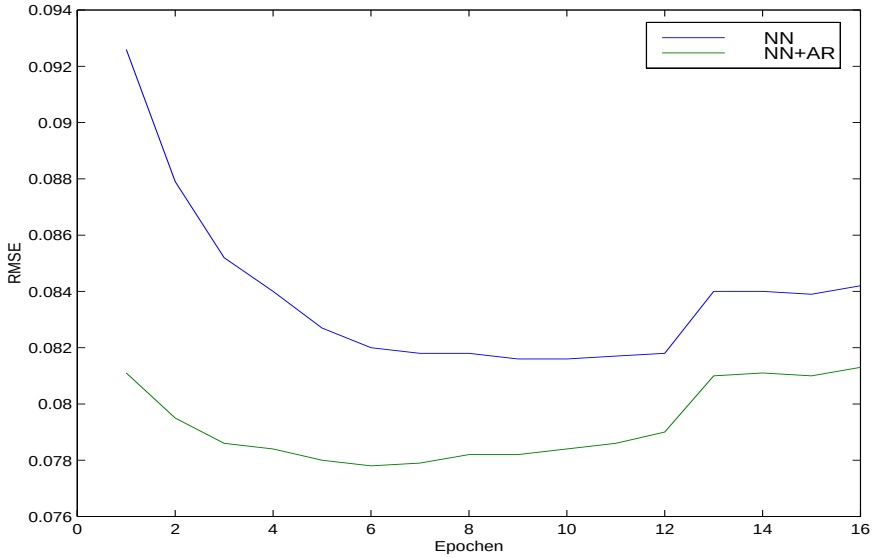


ABBILDUNG 42: Fehler mit und ohne Fehlerextrapolation bei unterschiedlicher Epochenanzahl

Ergebnisvergleich mit und ohne Vortraining

Bei dem sehr guten Ergebnis des neuronalen Netzes mit Fehlerextrapolation bleibt zu untersuchen, welchen Einfluss das Vortraining mit den ersten 20000 Daten auf das Ergebnis hat. Zum Vergleich wurde kontinuierliches Lernen durchgeführt, ohne jegliche Initialisierung. Die Fensterbreite war wieder 10, die Lernrate 0.0005, und mit 6 Epochen pro Fenster wurde trainiert.

	T	U	N	E	UE
Anzahl	249020	117007	46742	10683	5341
Neuronales Netz mit V.	0.0824	0.0918	0.0811	0.1138	0.1151
Neuronales Netz + AR m.V.	0.0782	0.0899	0.0740	0.1019	0.1137
Neuronales Netz ohne V.	0.0875	0.0972	0.0866	0.1182	0.1185
Neuronales Netz + AR o.V.	0.0827	0.0949	0.0787	0.1061	0.1168

TABELLE 19: RMSE für Testdaten, mit und ohne Vortraining

Die Ergebnisse ohne Vortraining sind um ca. 5 % schlechter, nur auf den Testdaten gemessen. Ohne Vortraining dauert es signifikant länger, bis das Netz den funktionalen Zusammenhang gelernt hat. In der folgenden Abbildung ist illustriert, wie sich die Fehler über die Zeit entwik-

keln. Die mittlere Kurve stellt den geglätteten Fehler des Neuronalen Netzes *ohne*, die untere Kurve *mit* Vortraining dar. Selbst nach 269000 Datensätzen besitzt die Version ohne Vortraining noch nicht die Performance der vortrainierten Variante. Die oberste Kurve zeigt zum Vergleich den geglätteten Fehler einer Linearen Regression, angepasst jeweils mit den letzten 600 Daten.

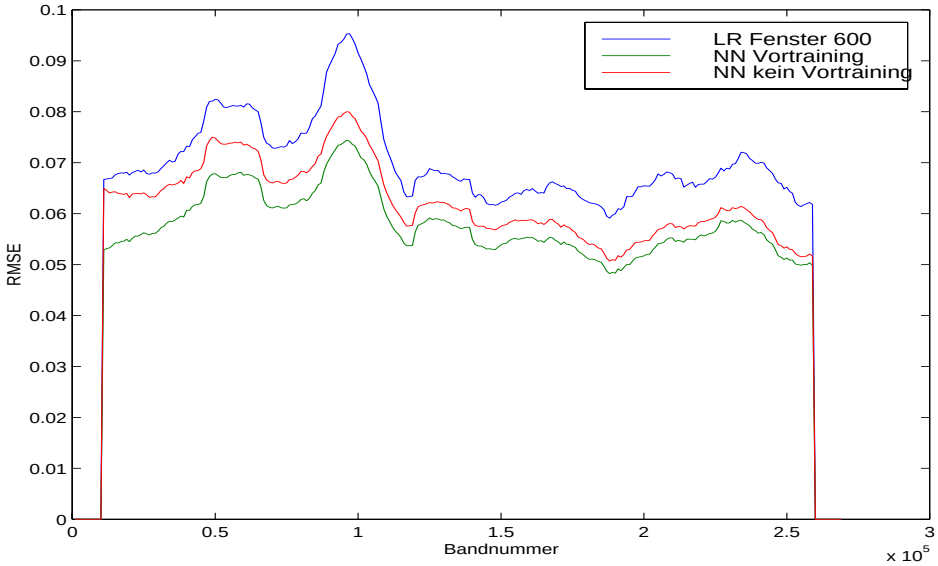


ABBILDUNG 43: Fehlerverlauf mit und ohne Vortraining

5.2.3 Adaptive Raumaufteilung

Ergebnisse für den Benchmark-Datensatz B

Es wird nun das auf S. 17f beschriebene Verfahren zur automatischen Raumaufteilung auf den Datensatz B angewendet. Der Benchmark-Datensatz B ist ein den an einer Walzanlage herrschenden Verhältnissen nachempfundener synthetischer Datensatz. Er besteht aus 100.000 Datensätzen mit 11 Eingangsvariablen und einem Ausgang. Zum Vergleich wurde zunächst das beste Ergebnis bestimmt, das mittels eines Neuronalen Netzes mit Vortraining auf den ersten 10000 Datensätzen und anschließendem kontinuierlichen Lernen erzielt werden konnte. Die besten unter den betrachteten Einstellungen waren durch eine Lernrate von 0.0005, durch 5 Neuronen im hidden layer und durch eine Fensterbreite von 1 für das Nachtrainieren gegeben. Es wurde der Backpropagation-Algorithmus verwendet. Wiederum wurde mit Hilfe mehrerer AR-Modelle (jeweils ein AR(1)-Modell für Umstellungsbänder bzw. die ersten Nachfolgebänder, sowie ein AR(2)-Modell für die weiteren Nachfolgebänder) der Fehler extrapoliert.

	T	U	N	E	UE
Anzahl	90000	15393	15111	4606	798
Neuronales Netz	0.1639	0.1676	0.1671	0.1827	0.1912
Neuronales Netz + AR-Modelle	0.1242	0.1675	0.1217	0.1338	0.1916

TABELLE 20: RMSE auf den 90.000 Testdaten nach Vortraining

Zur Messung der Performance des Ansatzes mit der automatisch generierten Norm wurden diverse Einstellungen betrachtet und variiert. Die Initialstützstellen wurden durch zufälliges Ziehen aus den ersten 10000 Datensätzen gewonnen. Da die Anfangswerte der Stützstellen das Endergebnis beeinflussen, wurde dieses Verfahren wiederholt durchgeführt, um Aussagen über die Stabilität zu gewinnen. Gleichermaßen lässt sich ein Modell gewinnen, dass die Mittelwerte der einzelnen Prognosen der verschiedenen Durchläufe verwendet. Dieses Modell sollte verbesserte Prognosen liefern können. In der Tabelle 21 sind zunächst die Ergebnisse für 10 Durchläufe aufgelistet. Die rechte Seite zeigt die Verbesserung der Resultate mit Extrapolation der Fehler durch die AR-Modelle. In jeder zweiten Zeile steht das Ergebnis, falls der Mittelwert der bisherigen Prognosen verwendet wird (Die Werte in der Zeile „*Gesamt i“ stellen den Fehler der Mittelwertprognose der Durchläufe 1 bis i dar.) Die Werte der Parameter betrugen $\lambda = 0,005$, $\varepsilon = 0,002$ und es wurden 20 Stützstellen verwendet.

	T	U	N	E	UE	T	U	N	E	UE
Anzahl	90000	15393	15111	4606	798	90000	15393	15111	4606	798
Durchlauf 1	0.2128	0.2167	0.2132	0.2503	0.2568	0.1378	0.2156	0.1266	0.1574	0.2553
Durchlauf 2	0.2446	0.2502	0.2466	0.2732	0.2738	0.1569	0.2491	0.1444	0.1678	0.2724
*Gesamt 2	0.2126	0.2161	0.2140	0.2406	0.2450	0.1361	0.2148	0.1243	0.1482	0.2434
Durchlauf 3	0.2341	0.2366	0.2363	0.2701	0.2757	0.1591	0.2353	0.1518	0.1802	0.2748
*Gesamt 3	0.2100	0.2123	0.2115	0.2379	0.2420	0.1347	0.2110	0.1231	0.1476	0.2404
Durchlauf 4	0.2250	0.2322	0.2270	0.2631	0.2657	0.1550	0.2312	0.1490	0.1655	0.2642
*Gesamt 4	0.2054	0.2080	0.2070	0.2337	0.2366	0.1324	0.2068	0.1214	0.1432	0.2351
Durchlauf 5	0.2305	0.2333	0.2324	0.2991	0.2973	0.1450	0.2322	0.1304	0.1762	0.2954
*Gesamt 5	0.2029	0.2054	0.2046	0.2330	0.2352	0.1298	0.2042	0.1184	0.1413	0.2336
...	...	...	...	...	...	...	...	...	...	...
Durchlauf 10	0.2157	0.2184	0.2175	0.2590	0.2613	0.1376	0.2172	0.1254	0.1581	0.2592
*Gesamt 10	0.2026	0.2048	0.2043	0.2305	0.2325	0.1283	0.2034	0.1160	0.1393	0.1393

TABELLE 21: RMSE bei 20 Stützstellen, rechte Seite mit Fehlerextrapolation.

Die Ergebnisse zeigen, dass zumindest der Mittelwert aus mehreren fehlerextrapolierten Prognosen den Vergleich zur fehlerextrapolierten Netzprognose nicht scheuen muss. Es soll noch einmal der Vergleich zur einfachen linearen Regression gezogen werden, um zu sehen, wieviel die Aufteilung des Arbeitsraumes und die Verwendung der automatisch generierten Norm überhaupt zur Fehlerreduktion beiträgt.

	T	U	N	E	UE	T	U	N	E	UE
Anzahl	90000	15393	15111	4606	798	90000	15393	15111	4606	798
bestes Modell	0.2157	0.2184	0.2175	0.2590	0.2613	0.1376	0.2172	0.1254	0.1581	0.2592
schlechtestes Modell	0.2562	0.2599	0.2565	0.3075	0.3036	0.1676	0.2590	0.1549	0.1986	0.3021
*Gesamt 10	0.2026	0.2048	0.2043	0.2305	0.2325	0.1283	0.2034	0.1160	0.1393	0.1393
Lineare Regression	0.3341	0.3331	0.3347	0.3469	0.3372	0.1661	0.3323	0.1122	0.1681	0.3349

TABELLE 22: Vergleich der RMSE-Fehler für die Testdaten. Rechte Seite mit AR-Fehlerextrapolation

Aus der Tabelle wird ersichtlich, dass die Raumaufteilung bei den reinen Regressionen ohne Fehlerextrapolation selbst im schlechtesten Durchlauf deutliche Vorteile bringt. Signifikant ist, dass die Regression mit Fehlerextrapolation ohne Aufteilung bei Umstellungsbändern deutlich schlechter abschneidet, aber die besten Ergebnisse für das erste Nachfolgeband ergibt.

Raumaufteilung mit quadratischen Modellen

Anstelle der linearen Modelle pro Stützstelle können auch andere Modelle verwendet werden. Relativ einfach zu handhaben sind Modelle, die zusätzlich zu den eigentlichen Variablen noch deren Quadrate verwenden. Mit den gleichen Einstellungen und den gleichen Anfangsstützstellen wie im letzten Abschnitt ergibt sich die folgende Ergebnistabelle 23: Wiederum ist das schlechteste Ergebnis mit Raumaufteilung immer noch deutlich besser als das globale lineare Modell.

	T	U	N	E	UE	T	U	N	E	UE
Anzahl	90000	15393	15111	4606	798	90000	15393	15111	4606	798
bestes Modell	0.1822	0.1924	0.1849	0.2030	0.2161	0.1354	0.1912	0.1333	0.1428	0.2144
schlechtestes Modell	0.1952	0.2008	0.1988	0.2233	0.2318	0.1345	0.1995	0.1274	0.1465	0.2300
*Gesamt 10	0.1771	0.1808	0.1806	0.1991	0.2051	0.1234	0.1794	0.1175	0.1319	0.2034
Lineare Regression mit Quadraten	0.2855	0.2856	0.2868	0.3037	0.2965	0.1511	0.2847	0.1136	0.1560	0.2950

TABELLE 23: RMSE bei 20 Stützstellen, rechte Seite mit Fehlerextrapolation.

5.2.4 Raumaufteilung mit flexiblen Metriken

Ergebnisse für den Datensatz A

Das im Abschnitt “Raumaufteilung mit flexiblen Metriken” auf Seite 20 hergeleitete Verfahren zur Raumaufteilung mit flexiblen Metriken soll zum Schluss wieder am Datensatz A getestet werden. Zunächst wird das Verfahren aus “Automatisch generierte Metrik” auf Seite 17 betrachtet, das im Falle des künstlichen Benchmark-Datensatzes zu einer erheblichen Verbesserung der Prognosegüte geführt hatte. Als Alternativ-Verfahren wird zunächst wieder die einfache Aufteilung mittels der euklidischen Norm betrachtet. In beiden Fällen ist jeweils nur ein Modell für die Modellprognose verantwortlich. Es findet also keine Verschmelzung benachbarter Modellprognosen statt. Auch wird die Modellgleichung eines einzelnen Modells über eine einfache und nicht über eine gewichtete Regression gewonnen. Die Tabelle zeigt die erzielten Resultate:

	T	U
Anzahl	249020	117007
Neuronales Netz + AR	0.0782	0.0899
Sigmanormiert eukl.	0.0924	0.1062
Autonorm	0.0914	0.1052

TABELLE 24: RMSE-Ergebnisse des besten Netzes, ohne Trainingsdaten

Der Performancegewinn durch Verwendung von Autonorm gegenüber der eukl. Norm ist beim realen Datensatz minimal, und das gute Ergebnis des globalen Neuronalen Netzes kann nicht erreicht werden. Diese Konstellation war der Ausgangspunkt für eine Reihe von Erweiterungen, die im Algorithmus des Verfahrens der Raumaufteilung mit flexiblen rezeptiven Feldern aufgenommen wurden:

	T	U
Anzahl	249020	117007
Flexible Metrik	0.0899	0.1028
+ Offsetanpassung	0.0863	0.0996
+ Verschmelzen	0.0795	0.0912
+ Vergessensfaktor	0.0768	0.0890
3x parallel Flexible Metrik	0.0737	0.0848
10x parallel Flexible Metrik	0.0723	0.0828
30x parallel Flexible Metrik	0.0716	0.0819

TABELLE 25: RMSE-Ergebnisse des besten Netzes, ohne Trainingsdaten

Im Falle des echten Datensatzes schneidet der reine Ansatz mit flexiblen Metriken besser ab als das Verfahren "Autonorm". Eine weitere Verbesserung bringt der dem eigentlichen Modellupdate vorangehende Schritt der Offsetanpassung. Wurde bis zu diesem Zeitpunkt noch nicht die

Performance des Neuronalen Netzes erreicht, so ändert sich dies mit dem Verschmelzen benachbarter Modellprognosen gemäß der Aktivierung der zugehörigen rezeptiven Felder und der Einführung des Vergessensfaktors. Hier wird zum ersten Mal der Fehler des globalen Modells unterboten.

Da die Ausgangslagen der rezeptiven Felder zufällig erzeugt werden, bietet sich die wiederholte Anwendung mit anderen zufällig erzeugten Stützstellen an. Die Berechnung kann natürlich parallel ausgeführt werden. Als Funktionsapproximation wird einfach der Mittelwert aller Durchläufe genommen. Bei geringerer Rechenzeit im Vergleich zum Neuronalen Netz wird letztendlich bei 30 Durchläufen ein um 10 Prozent verminderter Fehler erzielt.

Zusammenfassung

Im Mittelpunkt der Untersuchungen stand die Frage, in welcher Form die Identifizierung eines zeitvarianten Systems geschehen sollte. Für den Problembereich der Walzwerksteuerung, speziell der Endprofilprognose, wurden zunächst einfache globale lineare Modelle betrachtet, die sich vor allem durch Schnelligkeit auszeichnen. Eine erste Verbesserung zur Berücksichtigung der Zeitvarianz wurde durch den Einsatz von Fenstern erreicht; das globale lineare Modell wird dabei nur mit aktuellen Werten der jüngsten Vergangenheit gefüttert.

Unter der bisher getroffenen Annahme, dass nicht die Nichtlinearität der Funktion, sondern die Zeitvarianz das größere Problem ist, können auch Differenzen der Eingänge und Ausgangswerte aufeinanderfolgender Bänder betrachtet werden. Oftmals kann so der Einfluss einer nichtstationären Störung gemindert oder sogar beseitigt werden.

Einen entscheidenden Schub in der Verbesserung der Modellanpassung gab die Beobachtung, dass bei den bisher verwendeten Verfahren die Zeitreihe der Fehler noch Struktur besitzt. Schon mit klassischen Verfahren, speziell den ARIMA-Methoden nach Box-Jenkins, konnten die Prognoseergebnisse deutlich verbessert werden, indem durch Berücksichtigung der extrapolierten Fehler die eigentliche Prognose korrigiert wurde. Es stellte sich auch heraus, dass auf die Verwendung von Fenstern sowohl für die eigentliche Zeitreihe der Profile, als auch von Fenstern für die Fehlerreihe (zur schnelleren Adaption der Parameter der ARIMA-Modelle) verzichtet werden konnte, falls die Fehler mittels einer ARIMA-Zeitreihe modelliert wurden.

Einfache "mixture of experts"-Systeme wurden hergeleitet, welche zu verschiedenen Zeitpunkten mit unterschiedlichen Prognosehorizonten gemachte Prognosen über einen Regressionsansatz kombinieren. Auch diese Variante ergab gegenüber einem einfachen Modell eine signifikante Verbesserung.

Die bisherigen Modelle berücksichtigen hauptsächlich die Zeitvarianz und vernachlässigen etwas den möglicherweise nichtlinearen Modellzusammenhang. Neuronale Netze bieten die Möglichkeit, in relativ einfacher Form diese Nichtlinearitäten zu modellieren, wobei natürlich das Problem der Zeitvarianz weiterhin bestehen bleibt. Eine Vielzahl von Architekturen und Lernverfahren wurde ausgetestet, die Zeitvarianz wurde durch die Verwendung von Fenstern berücksichtigt. Die Netze wurden auf einer größeren Menge von Daten (beim Datensatz A beispielsweise mit 20000 Daten) vortrainiert und dann kontinuierlich mit den jeweils letzten Datensätzen nachtrainiert, wobei der Parameter der Fensterbreite ebenfalls variiert wurde. Der

bemerkenswert geringe Fehler, der dabei in Verbindung mit der ARIMA-Fehlerextrapolation erzielt wurde, ergab lange Zeit eine untere Schwelle, die von späteren Verfahren zunächst nicht untertroffen wurde.

Bemerkenswert war dabei die Tatsache, dass das Vortraining entschieden wichtiger war als das Nachtrainieren. Wurde auf das Nachtrainieren verzichtet, so ergab sich annähernd die gleiche Prognosegüte, aber nur dann, wenn die ARIMA-Fehlerextrapolation eingesetzt wurde. Dies mag als Hinweis interpretiert werden, dass sich die Zeitvarianz hauptsächlich durch eine Verschiebung des Offsets ausdrückt, und weniger durch eine Verformung der Funktion. Nach 20000 Datensätzen zum Vortrainieren war möglicherweise die Form der Funktion genügend genau angepasst.

Die Idee war nun, lineare Modelle (schnelles Update durch Rekursionsformeln) mit der Fähigkeit zur Approximation nichtlinearer Funktionen zu versehen. Dazu wurde der Eingangraum aufgeteilt, sodass nun eine größere Anzahl von linearen Modellen verwendet wurde, wobei zunächst jedes Modell für einen klar abgegrenzten Bereich verantwortlich war. Die Grenzen zwischen zwei lokalen Modellen sollten jedoch adaptierbar sein. Anschaulich gesehen sollte jedes lokale Modell sich in Gebiete ausdehnen, in denen seine Prognosen besser waren als die anderen, und aus denjenigen zurückziehen, in denen die Prognosen den anderen bezüglich der Güte unterlegen waren.

Für einen gegebenen Eingangsvektor wird ein Maß benötigt, das das verantwortliche Modell für die Prognose gemäß des geringsten Abstandes ermittelt. Der erste Ansatz, "Autonorm", verwendete nur globale Informationen für dieses Abstandsmaß. Eingangsvariablen, die weniger wichtig waren, wurden in diesem Maß weniger berücksichtigt. Auf einem künstlichen Datensatz ergab dieses Verfahren eine deutliche Verbesserung gegenüber dem normalen euklidischen Abstandsmaß.

Schließlich wurde ein Verfahren entwickelt, das den Eingangsraum auf etwas komplexere Art und Weise aufteilt. Durch die Verwendung von flexiblen Aktivierungsfunktionen ist es möglich, den Einflussbereich eines Modells stetig zu ändern. Das Ausmaß der Änderung wird allein durch lokale Eigenschaften beeinflusst, sodass die Stabilität an anderen Arbeitspunkten im Eingangsraum gewährleistet ist. Außerdem ist die Möglichkeit gegeben, die Prognosen verschiedener Modelle gemäß ihrer Aktivierungen zu gewichten und so selbst mit linearen Modellen eine stetige Approximation zu erreichen.

Die Einführung von Vergessensfaktoren in die Modellupdates der linearen Modelle, sowie eine vorgeschaltete Offsetanpassung ergab zum ersten Male eine Verbesserung gegenüber dem globalen Neuronalen Netz. Durch parallele Verarbeitung mehrfacher Durchläufe des mit zufälligen Lageparametern startenden Verfahrens konnte der Fehler auf einen Wert gesenkt werden, der etwa 10 Prozent unter dem Fehlerwert des globalen Netzes lag, und dies noch bei geringerer Rechenzeit.

Erwähnenswert ist die Tatsache, dass alle Verfahren ohne Kenntnis eines physikalischen Modells hergeleitet wurden. Selbst die Bedeutung der Eingangsvariablen war aus Gründen der Geheimhaltung nicht bekannt. Deshalb war es auch nicht einfach, Bereiche für Ausreisser festzulegen, da durchaus Variablen mit sehr unterschiedlichen gültigen Wertebereichen vorliegen können. Trotzdem gelang es, Prognosegüten zu erzielen, die vergleichbar waren mit den

von SIEMENS erzielten Werten, obwohl beim Partner das technische Know-How für die Walzanlage vorlag. Es ist deshalb anzunehmen, dass die hergeleiteten Verfahren mit geringem Aufwand auf andere Situationen, bei denen die Identifikation zeitvarianter Systeme im Vordergrund steht, übertragen werden können.

5.3 Prognose der Zwischenprofile

5.3.1 Analytische Modellbildung

Die Aufgabenstellung bei der Vorhersage der Zwischenprofile ist in Abb. 39 zusammengestellt. Die verschiedenen an einem Gerüst auftretenden Teilprozesse sind stark ineinander verzahnt. Mit Ausnahme der am Ein- und Ausgang der Walzstraße messbaren Größen, der einzustellenden Kräfte und der Durchmesser der Arbeitswalzen werden alle für die Zwischenprofilvorhersage relevanten Größen aus einer Voraus– bzw. eine Nachberechnung, d.h., über weitere Modelle, erhalten. Die Vorausberechnung prognostiziert die jeweilige Prozessgröße anhand gegebener Einstellungen der sie beeinflussenden Größen, die Nachberechnung korrigiert diese Werte entsprechend einer Differenz in der von ihnen beeinflussten Größen zu deren Istwert.

Profildefinition

Das in den Datensätzen verzeichnete Profil entsprechend der Definition

$$p = h_M - \frac{h_L + h_R}{2} \tag{1}$$

ist proportional dem negativen Koeffizienten des quadratischen Terms (a) bei quadratischer Interpolation der Banddicken h in der Mitte (h_M), am linken (h_L) bzw. rechten (h_R) Rand des Bandes, die Horizontale als Symmetrieachse vorausgesetzt (B: Bandbreite):

$$\begin{aligned} h &= h(x) = ax^2 + bx + c \\ \frac{h_L}{2} &= h(0),\ \frac{h_M}{2} = h\left(\frac{B}{2}\right),\ \frac{h_R}{2} = h(B) \\ a &= \frac{1}{B^2}[-2h_M + h_L + h_R] = -\frac{1}{2B^2} \cdot p \end{aligned} \tag{2}$$

Da die Bandbreite unberücksichtigt bleibt, beschreibt das Profil nicht die Form der Bandoberfläche, sondern lediglich den Dickenunterschied zwischen Bandmitte und Rand, gemittelt über die linke und die rechte Bandhälfte. Von zwei Bändern gleichen Profils ist dasjenige geringerer Breite stärker gewölbt.

Bei konvexen Oberflächen wird das Profil positiv, bei konkaven negativ. Es wird Null, wenn die drei Werte auf einer Geraden liegen, unabhängig davon, welchen Anstieg diese Gerade besitzt. Somit werden stark unterschiedliche Formen nicht unterschieden: ein Profilwert von Null kann einen rechteckigen, dreieckigen oder trapezförmigen Bandquerschnitt beschreiben. In der Praxis werden symmetrische Profile (von der Bandmitte aus gesehen) angenommen, so dass dies kein Problem darstellt.

Modellgleichungen und Parameter

Profilbestimmung und -steuerung gehen davon aus, dass beim Durchgang eines Bandes durch ein Gerüst das Walzspaltprofil (die Kontur der Arbeitswalzen im Walzspalt) komplett auf das Band übertragen wird: das Austrittsprofil des Bandes ist identisch dem Walzspaltprofil [36]. Die korrekte Bestimmung des Walzspaltprofils setzt die Kenntnis der Walzkraftverteilung über die Ballenlänge voraus. In Modellen wie [36] wird sie für ein Vier–Walzen–Gerüst über einen Iterationsmechanismus bestimmt:

1. Initialisierung auf konstante Walzkraftverteilung
2. Walzkraftverteilung ➔ Elastische Verformung des Walzensatzes ➔ Kontur/Profil des auslaufenden Bandes
3. Ein- und auslaufende Banddickenverteilung ➔ Verteilung der Zugspannungen im auslaufenden Band
4. Ein- und auslaufende Banddickenverteilung, Zugspannungsverteilung, Fließkurve, Reibungszahl ➔ Walzkraftverteilung; weiter mit 2. bis Konvergenz der Walzkraftverteilung.

Das Walzspaltprofil selbst wird aus einer weiteren Iterationsroutine erhalten, die für die ermittelte Walzkraftverteilung zunächst Biegung und Abplattung der Arbeitswalzen bestimmt. Für zwei benachbarte Walzen (beim Vier–Walzen–Gerüst: Arbeits- und Stützwalze) gilt, dass sie sich in der gemeinsamen Kontaktzone so verformen, dass die entstehenden Kraftverteilungen zu äußeren Kräften und Momenten im Gleichgewicht stehen. Die Verformungen müssen geometrisch kompatibel sein – Walzen können sich nicht durchdringen. Ist diese Bedingung nicht eingehalten, wird die Abplattung korrigiert und eine neue Verteilung der Zwischenkraft zwischen Arbeits- und Stützwalze bestimmt.

Das Walzspaltprofil ergibt sich aus der Superposition von Biegelinie, Abplattung und Schliff der Arbeitswalze. Das Modell [36] ist für einen Kaltwalzvorgang aufgestellt worden; beim Warmwalzen wird zusätzlich ein Temperaturmodell nötig.

Ist eine dieser Komponenten des Walzspaltmodells nur unzureichend bekannt, kann sich eine starke Abweichung zum real vorhandenen Walzspalt ergeben – Austrittsprofil des Bandes und Walzspaltprofil stimmen nicht überein.

Der Industriepartner Siemens ANL verwendet daher eine empirische Übertragungsfunktion, die diese Abweichungen durch Einbeziehung des einlaufenden Bandprofiles zu kompensieren sucht. Diese Vorgehensweise ist jedoch strenggenommen nur dann zulässig, wenn es eine Beziehung zwischen den Abweichungen der im Modell verwendeten Einflussgrößen zu deren Realwerten und dem Eintrittsprofil des Bandes gibt.

Die Profilvorausberechnung für ein Profil p_n am Ausgang des Gerüstes n erfolgt anhand der Gleichung

$$p_n = k_n \cdot \frac{h_n}{h_{n-1}} \cdot p_{n-1} + (1 - k_n) \cdot W_n + c_{n3} \cdot a \cdot h_n , \qquad (3)$$

(3) setzt sich aus einer Linearkombination der Profile W_n (Profil des Walzspaltes) und p_{n-1} (Banddickenprofil) und einem adaptiven Term zusammen. Den Grad des Einflusses beider Profile in der Linearkombination bestimmt der Faktor k_n. Er ist eine nichtlineare Funktion, die ne-

ben den Prozessvariablen bzw. Parametern der Gerüste (Arbeitswalzenradius D, Banddicke h und Bandbreite B) zwei Koeffizienten c_{n1} und c_{n2} enthält, die zusätzliche Variabilität in die Übertragungsfunktion der Gerüste einbringen können.

Der Adaptionsterm $c_{n3} \cdot a \cdot h_n$ verwendet den Adaptionskoeffizienten a, der aus einer Folge vorangegangener Endprofile berechnet wird und für aufeinanderfolgende Bänder verschieden ist.

Optimale Koeffizienten und Modellgüte

Um eine Übersicht über die Approximationseigenschaften des analytischen Modells bezüglich des Endprofils zu gewinnen, wurden die Koeffizienten c unter Verwendung der MATLAB `leastsq`-Routine [37] optimiert.

Im Vergleich von mehreren Testdatensätzen ist zu erkennen, dass signifikant unterschiedliche Orte im sechsdimensionalen Raum der zu optimierenden Koeffizienten c_B ähnliche Gütewerte erzeugen. Die Gütefunktion

$$R(\boldsymbol{c}, \boldsymbol{X}) = \sum_{s=1}^{S} (\hat{p}_{s,7}(\boldsymbol{x}_s, \boldsymbol{c}) - p_{s,7})^2, \tag{4}$$

die der Optimierung zugrundeliegt, ist einerseits von der Gesamtheit der Daten, $\boldsymbol{X}$, andererseits von den Modellkoeffizienten $\boldsymbol{c} = \{c_1, c_2, c_3, c_D, c_h, c_B\}$ abhängig und zeigt eine zerklüftete Struktur. Diese wird sichtbar in der Variation der Koeffizienten c_1 und c_B in Abbildung Abb. 44. Dort sind weitere wesentliche Abhängigkeiten im Arbeitspunkt dargestellt.

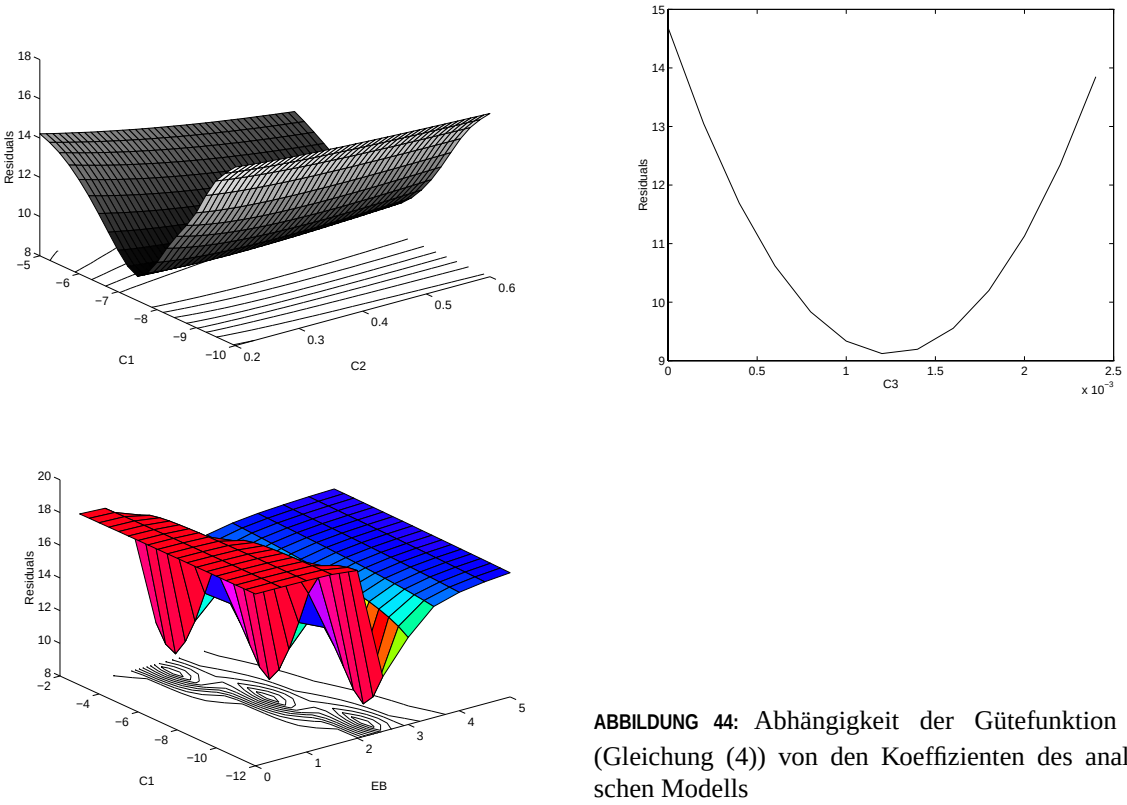


ABBILDUNG 44: Abhängigkeit der Gütefunktion R (Gleichung (4)) von den Koeffizienten des analytischen Modells

Sensitivitätsanalyse

In einer Sensitivitätsanalyse des analytischen Modells sollten folgende Punkte geklärt werden:

1. Wie reagiert das Modell auf leicht veränderte Eingangswerte? (Sensitivität bezüglich der Daten)
2. Welcher Fehler entsteht in den Zwischenprofilen, wenn in der Modellierung (z.B. durch ein künstliches neuronales Netz) für das Endprofil ein maximaler Fehler zulässig ist? Die Sensitivität des analytischen Modells bezüglich der Koeffizienten kann diese Frage beantworten unter der Annahme, dass das analytische Modell bereits die beste Prozessbeschreibung gibt und ein neu an die Daten zu adaptierendes Modell diesem Modell qualitativ sehr nahe kommt.

Sensitivität bezüglich der Daten

Die Sensitivitäten, d.h. die partiellen Ableitungen des Ausganges $\hat{p}_7^{AM}$ des analytischen Modells nach den einzelnen Prozessvariablen im Arbeitspunkt, sind einerseits für die worst–case Situationen (Verwendung der Maxima und Minima der Prozessvariablen (Banddicken, Bandbreiten, Arbeitswalzendurchmesser) über die Gesamtheit der Daten), andererseits für die tatsächlich eingetretenen Situationen ermittelt worden. Abb. 45 stellt diese Werte zusammen. Die Sensitivitäten nehmen generell sehr niedrige Werte an; ein eindeutiger Trend zu höheren Sensitivitäten mit steigender Gerüstnummer ist nicht erkennbar. Eine Veränderung beispielsweise der Arbeitswalzendurchmesser um 200 mm im Gerüst 1 (das ist ca. das Vierfache seines Wertebereiches von 660 mm bis 715.35 mm) führt auf eine Veränderung von weniger als einem Mikrometer.

.

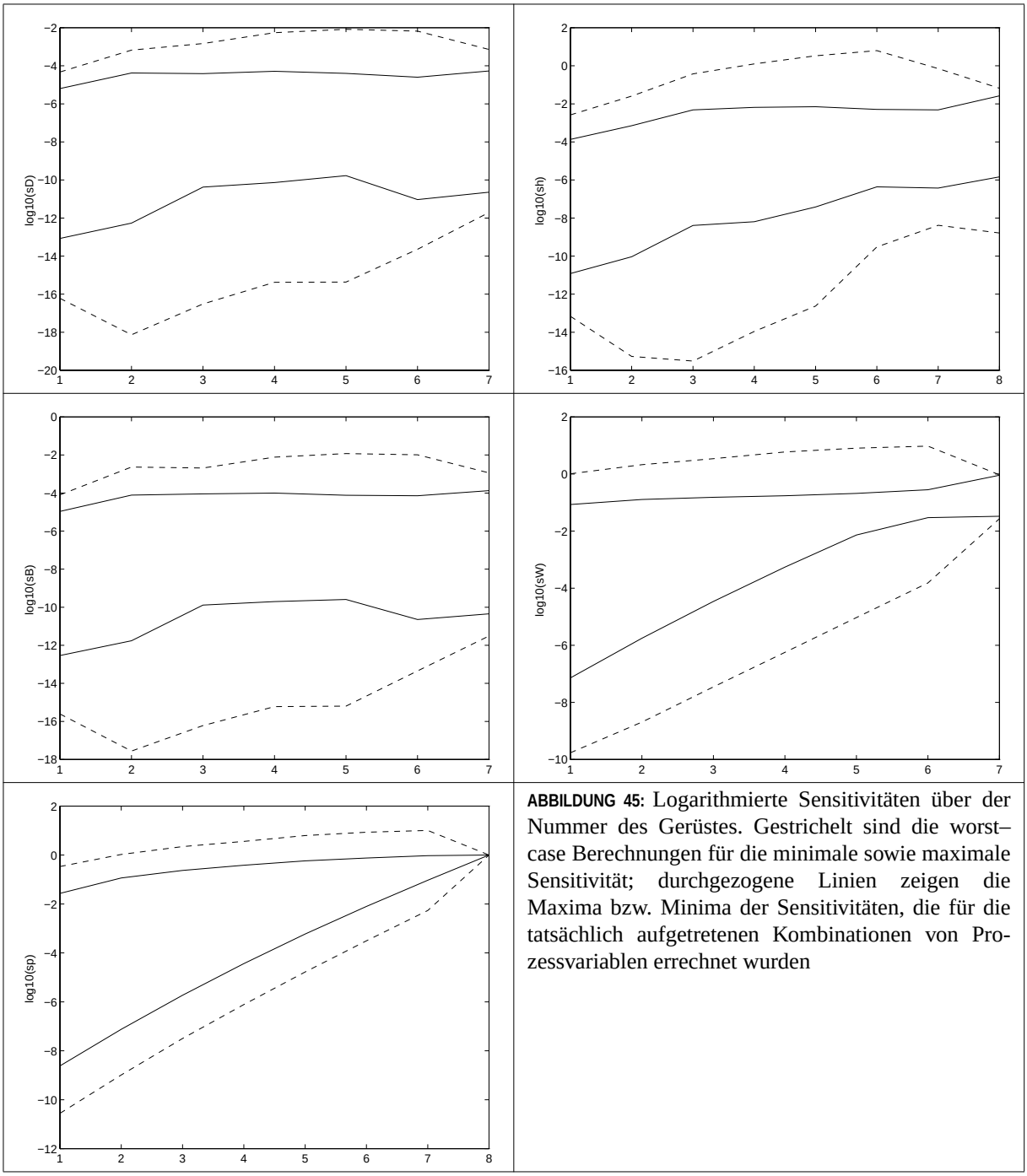


ABBILDUNG 45: Logarithmierte Sensitivitäten über der Nummer des Gerüstes. Gestrichelt sind die worst–case Berechnungen für die minimale sowie maximale Sensitivität; durchgezogene Linien zeigen die Maxima bzw. Minima der Sensitivitäten, die für die tatsächlich aufgetretenen Kombinationen von Prozessvariablen errechnet wurden

Sensitivität bezüglich der Koeffizienten

Wenn bei der Modellierung durch ein rekurrentes System am Ausgang ein definierter Approximationsfehler zulässig ist, dann kann der Fehler im Verlauf der Iterationen durchgängig vergrößert oder durchgängig verkleinert worden sein, oder positive und negative Änderungen können sich abwechseln. Es wird davon ausgegangen, dass jeweils ein Koeffizient des Modelles allein für den Approximationsfehler am Ausgang verantwortlich ist. Nimmt man an, dass sich der Fehler beim Durchgang durch die Gerüste nur in einer Richtung verändert (durchgängig wächst

oder durchgängig sinkt), gilt folgendes: Ist die Sensitivität des Ausganges $\hat{p}_7^{AM}$ bezüglich eines Koeffizienten betragsmäßig größer als die Sensitivität des nach dem ersten Gerüst erhaltenen Zwischenprofils $\hat{p}_1^{AM}$, erzeugt die Änderung dieses Koeffizienten eine kleinere Profiländerung nach dem ersten Gerüst; der Fehler des Zwischenprofile ist kleiner als der Fehler des Endprofils.

TABELLE 26: Sensitivität des analytischen Modells bezüglich der Koeffizienten

	c_1	c_2	c_3	e_D	e_h	e_B
p_7^{AM}	-0.141707	-0.122662	93.843965	0.910068	0.329288	-1.011624
p_1^{AM}	0.101204	0.091591	95.384450	-0.662234	-0.290103	-0.559284

In Tabelle 26 wurden die in den Daten aufgetretenen Kombinationen der Prozessvariablen im Arbeitspunkt verwendet und jeweils der betragsgrößte Wert der Signifikanz über alle Muster eingetragen. Strenggenommen müssten die partiellen Ableitungen aller Zwischenprofile nach den Koeffizienten berechnet werden; dies ist hier aus Aufwandsgründen nicht geschehen. Aus der Tabelle ist auch bei diesem Stand eine Aussage ableitbar: Die Sensitivität von $\hat{p}_1^{AM}$ ist für den Koeffizienten c_3 betragsmäßig größer als für $\hat{p}_7^{AM}$, dieser ist gleichzeitig derjenige mit dem größten Einfluss auf das Profil; damit muss im schlechtesten Fall angenommen werden, dass die Fehler der Zwischenprofile höher liegen als die Fehler des Ausgangsprofils. Der hohe Wert für c_3 bedeutet gleichzeitig, dass das Profil am Ausgang eines Gerüstes im wesentlichen durch die Adaption bestimmt wird und damit hauptsächlich von der Dicke des Bandes abhängt, das dieses Gerüst verlässt.

5.3.2 Neuronaler Ansatz

Das mathematische Modell des Walzvorganges zeigt sich in der Sensitivitätsanalyse als wenig befriedigend, die erreichte Genauigkeit resultiert im Wesentlichen aus der online Adaption der Koeffizienten. Es wurde bereits gezeigt, dass das Endprofil mit Neuronalen Netzen deutlich genauer vorhergesagt werden kann. Daher wird im folgenden versucht, auch das Problem der Zwischenprofile mit neuronalen Methoden anzugehen.

Die allgemeine mathematische Formulierung dieses Problems führt in das Gebiet der iterierten Funktionalgleichungen und erfordert die Berechnung von iterativen Funktionswurzeln:

Die gesamte Walzstraße werde durch eine Funktion $x_{out} = F(x_{in}, \bar{p}_1 \ldots \bar{p}_N)$ beschrieben, die aus den Eingangs-und Endprofilen x und den Prozess- und Materialparametern an jedem Gerüst $\bar{p}$ rekonstruierbar ist. Des weiteren ist bekannt, dass die Straße aus N identischen Gerüsten besteht, also F durch N-fache Verkettung einer Funktion $x_{i+1} = f(x_i, p_i)$ entsteht, $F = f^N$. Die Funktion f ist die gesuchte Übertragungsfunktion eines einzelnen Gerüstes, die zur Berechnung der Zwischenprofile x_i notwendig ist. $f = F^{1/N}$ bezeichtet man als N-te Funktionswurzel oder iterative Wurzel von F. Die analytische Berechnung solcher Funktionswurzeln stellt ein ausgesprochen kompliziertes mathematisches Problem dar [30] und es ergeben sich Fragen nach Existenz und Eindeutigkeit von Lösungen. In Abschnitt 4 wurde dargestellt, wie mit MLP's spezieller Topologie und abgestimmten Lernverfahren solche Funktionswurzeln berechnet werden können. Abb. 46 zeigt ein entsprechendes Netz, mit dem versucht wurde, den Zwischenprofilverlauf in einer 7-gerüstigen Walzstraße zu modellieren.

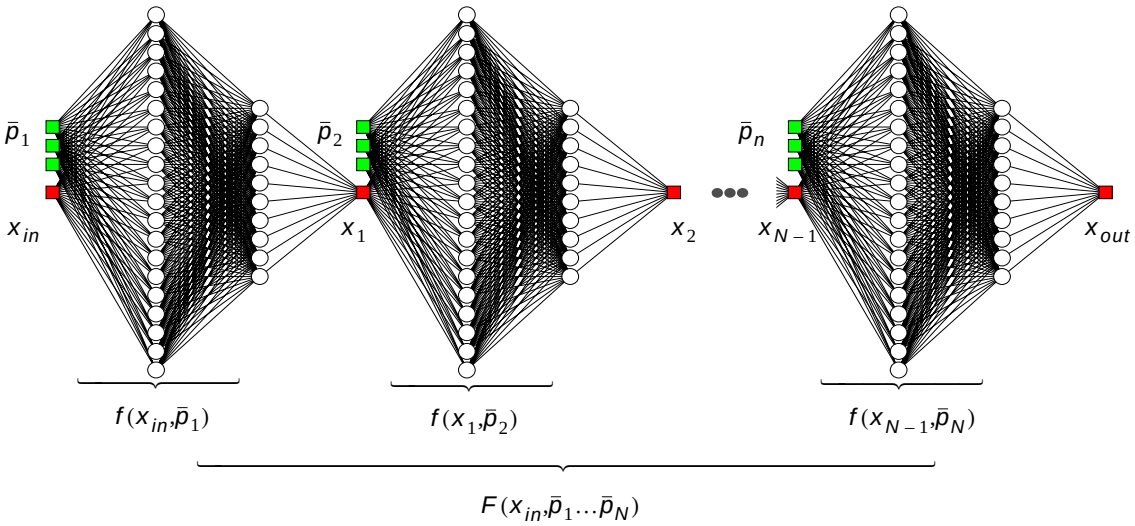


ABBILDUNG 46: Neuronales Netz, dessen Topologie einer Walzstraße entspricht Das gesamte Netz wird trainiert, das Endprofil vorherzusagen, also F approximiert die gesamte Strasse. Werden durch geeignete Massnahmen die Gewichte der Teilnetze identisch gehalten, approximiert ein Teilnetz ein einzelnes Gerüst f.

Experimente

Modellvariablen. Für die Belehrung eines rekurrenten mehrschichtigen Perzeptrons wurden zwei Sätze von Eingangsvariablen verwendet. Variante AM verarbeitet die gleichen Größen, wie sie auch das analytische Modell verwendet, Variante EXT nutzt einen erweiterten Variablensatz: das Walzspaltprofil, das selbst nur Ausgang eines weiteren Modells ist, wurde durch dessen (vermutete) Eingangsvariablen ersetzt. In einer Abwandlung der Variante AM, AMM, sind die Walzspaltprofile auf Null gesetzt worden, um deren Einfluss zu testen. Für die Belehrung wurden die Eingangsgrößen jeweils linear auf den Bereich [-1, 1], die Ausgangsgröße auf den Bereich [-0.9, 0.9] transformiert.

Netzgröße. Da unter den gegebenen Randbedingungen die Funktion des Einzelgerüstes nicht eindeutig rekonstruiert werden kann (Abschnitt 5), wurde das den jeweiligen Varianten entsprechende Netz mit der jeweils geringsten Zahl an Freiheitsgraden verwendet: die Netze besitzen 6 bzw. 11 Knoten in der Eingangsschicht (für die Variablen $p_{n-1}, h_{n-1}, h_n, r_n, B, W_n$ bzw. $p_{n-1}, h_{n-1}, h_n, r_n, B, F_{B,n}, F_{W,n}, F_{Z,n}, S_n, T_n, M, r_n = D_n/2$, keinen verdeckten Knoten und einen Ausgangsknoten mit nichtlinearer Transferfunktion. Ein solches Modell beschreibt bereits einen vollständigen Polynomansatz [38].

Netzstruktur. Neben dem rein neuronalen Ansatz (Abb. 46) wurde eine Variante getestet, die zur Einschränkung des Lösungsraumes einen Teil des analytischen Modells einbezieht. Abb. 47 zeigt das verwendete Hybridmodell – die Verknüpfung von Walzspaltprofil und einlaufendem Profil wird als prozessbeschreibend definiert; das Perzeptron übernimmt die Bestimmung des Koeffizienten dieser Linearkombination. Auch hier wurde das einfachste Netz mit drei Eingängen und einem nichtlinearen Ausgangsknoten ohne verdeckte Schicht verwendet.

Lernverfahren. Die Wichtungen der RMLP's wurden sowohl mit Gradientenabstieg (Backpropagation Through Time), als auch mit dem Verfahren nach Marquardt–Levenberg (MATLAB-Routine `leastsq`) bestimmt; das Hybridmodell wurde ausschließlich nach Marquardt–

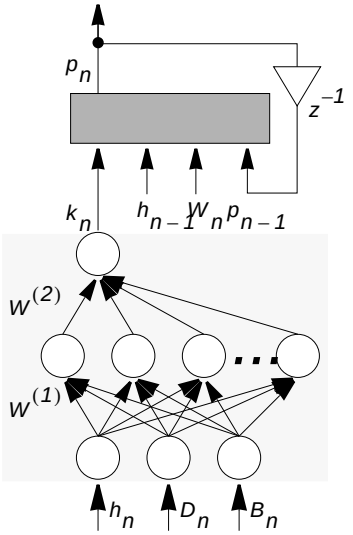


ABBILDUNG 47: Hybridmodell aus analytischem Modell und einer NN–Komponente. Ein mehrschichtiges Perzeptron wird hinsichtlich der Bestimmung der Koeffizienten k_n der Linearkombination ohne Verwendung des adaptiven Terms trainiert.

Levenberg konfiguriert. Der Gradientenabstieg sollte als erfolgreich beendet gelten, wenn die Güte des optimal auf die Trainingsmenge S_{trn} angepassten analytischen Modells, gemessen am betragsmaximalen Fehler ε_{max}^{AM} zwischen Vorgabe $p_{n,7}$ und Modellausgang $\hat{p}_{n,7}^{MLP}$ über alle Muster n, unterschritten wurde:

$$\max_{s \in S_{trn}} \left| p_{s,7} - \hat{p}_{s,7}^{AM,opt} \right| = \varepsilon_{max}^{AM} > \varepsilon_{max}^{MLP} = \max_{s \in S_{trn}} \left| p_{s,7} - \hat{p}_{s,7}^{MLP} \right| . \quad (5)$$

Bei Hinzunahme der Ausreißer ist dieses Kriterium jedoch zu schwach, so dass folgende verschärfte Bedingung Verwendung fand:

$$\operatorname{mean}_{s \in S_{trn}} \left(\left| p_{s,7} - \hat{p}_{s,7}^{AM,opt} \right| \right) + 3 \cdot \operatorname{std}_{s \in S_{trn}} \left(\left| p_{s,7} - \hat{p}_{s,7}^{AM,opt} \right| \right) = \varepsilon_{stat}^{AM} < \varepsilon_{max}^{MLP} = \max_{s \in S_{trn}} \left| p_{s,7} - \hat{p}_{s,7}^{MLP} \right| \quad (6)$$

Mean und Std bezeichnen Mittelwert bzw. Standardabweichung. In keiner der Simulationen wurde diese Bedingung für alle Muster erfüllt; die Anpassung der Wichtungen stoppte so beim Erreichen eines minimalen Fehlers auf der Testmenge.

Ergebnisse

In allen Experimenten lieferte das Verfahren nach Marquardt–Levenberg die beste Approximation an die Daten. In keinem Fall jedoch wurde die Güte des verwendeten bzw. des optimal angepassten analytischen Modells erreicht. Vergleicht man jedoch die erhaltenen statischen Modelle mit dem optimalen analytischen Modell ohne Adaptionsterm, dann sind hier in allen Experimenten, die den gesamten Datensatz zur Modellanpassung verwenden, durchgängig deutlich bessere Ergebnisse erreicht worden. Eine Kombination der erhaltenen Modelle mit einer Online–Adaption lässt erwarten, dass auch die Güte des optimal angepassten analytischen Modells mit Adaption überboten werden kann.

Die Approximationen mittels Backpropagation Through Time unterscheiden sich in der Initialisierung der Wichtungen und den Lernparametern (im wesentlichen Lernrate und Momentum–Term). Theoretisch zu erwarten, ist die Übertragungsfunktion des Einzelgerüstes bei vergleich-

baren Residuen nicht eindeutig rekonstruiert worden – in Abhängigkeit von der Initialisierung werden unterschiedliche Lösungen erhalten. Dies gilt ebenfalls für das Verfahren nach Marquardt–Levenberg.

Die Eliminierung der Walzspaltprofile zeigt keine wesentlichen Abweichungen zu den Versuchen, die die Walzspaltprofile einbeziehen. Dies deutet von seiten der Signalanalyse darauf hin, dass die Walzspaltprofile einen eher geringen Einfluss auf die erhaltenen Dickenprofile haben (dies widerspricht der Hauptannahme des analytischen Modells), andererseits könnte das verwendete Walzspaltmodell selbst unzulänglich sein. Der Ersatz dieser Größe durch deren (angenommene) Modellvariablen (Variablenmenge EXT) brachte keine Verbesserung: Die einfache Netzstruktur war nicht in der Lage, diese Modellierung mit zu übernehmen, bzw. die verwendeten Abhängigkeiten waren nicht adäquat gewählt. Das Hybridmodell bestätigt die Annahme der Linearkombination aus Walzspaltprofil und einlaufendem Profil ebenfalls nicht, hier wurden die schlechtesten Ergebnisse erreicht.

Abb. 48 zeigt die in einzelnen Experimenten erhaltenen Zwischenprofilverläufe für ein beliebig ausgewähltes Muster in der Gegenüberstellung zum analytischen Modell. Auffällig sind die Ausreißer bei den am Gerüst 7 gemessenen Austrittsprofilen, die sich nicht durch entsprechende “Anomalien” in jeweils einer anderen Prozessgröße erklären lassen. So können diese Werte durch Supressoreffekte [37] aus verschiedenen Prozessgrößen hervorgerufen werden, oder es handelt sich um zufällige Fehler bei der Profilmessung.

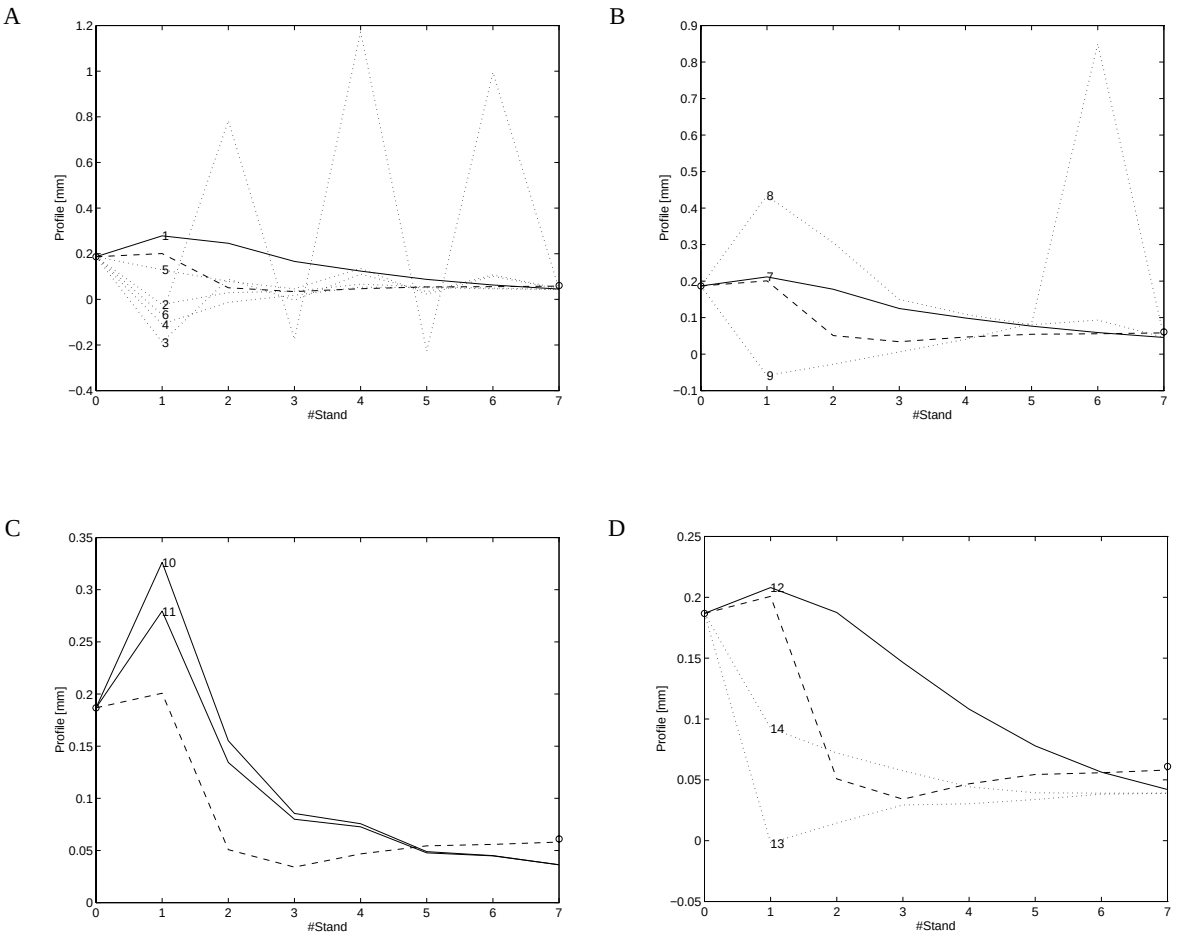


ABBILDUNG 48: Verläufe der Zwischenprofile für ein Beispielband. Gestrichelt: analytisches Modell; durchgezogene Linien: Optimierung nach Marquardt–Levenberg; gepunktet: Back-Propagation of Error. Die Werte auf der Ordinate zeigen das Profil am Ausgang des entsprechenden Gerüstes. Kreise: Ist–Profile. A-D: Verschiedene Initialisierungen

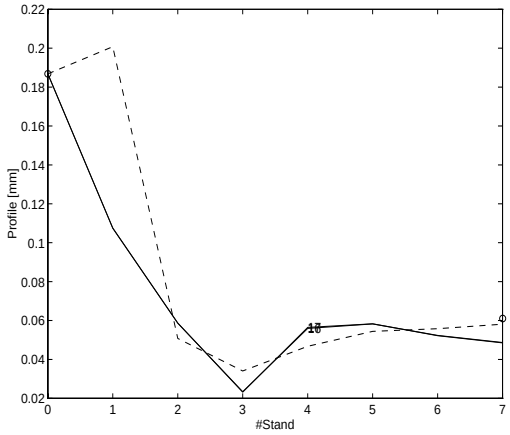


ABBILDUNG 49: Zwischenprofile aus dem Hybridmodell

.

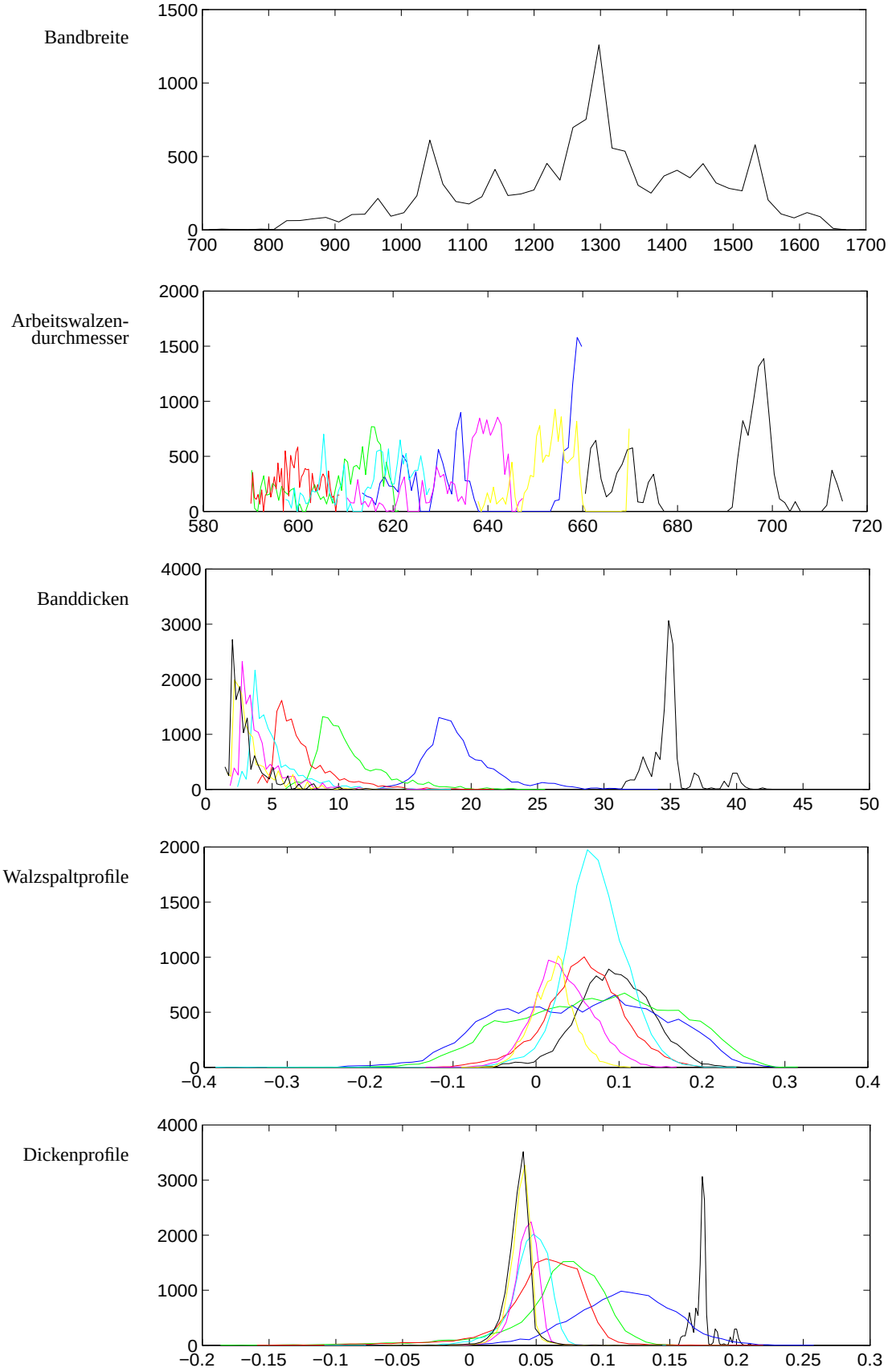


ABBILDUNG 50: Histogramme relevanter Prozessgrößen: Grauwerte unterscheiden die einzelnen Gerüste.

Zusammenfassung

Mit keinem der statischen RMLP's bzw. der Hybridmodelle wurde die Güte des adaptiven analytischen Modells erreicht. In allen Modellen, die mit dem kompletten vorhandenen Datenmaterial belehrt wurden, ist jedoch die Güte des optimalen statischen analytischen Modells stets unterboten worden. Die erreichten Güten kommen denen des adaptiven AM in der Mehrzahl der Fälle sehr nahe. Die Hinzunahme einer Adaption lässt hier deutliche Verbesserungen erwarten.

Von seiten der Fachexperten konnte keine Aussage zur Plausibilität der Zwischenprofilverläufe der Modelle im Vergleich zu den Zwischenprofilen aus dem analytischen Modell getroffen werden. Somit scheint es ohne weitere Anhaltspunkte bzw. Randbedingungen an die Verläufe der Zwischenprofile nicht sinnvoll, mit einer erweiterten Netzstruktur (Hinzunahme einer oder mehrerer Schichten verdeckter Knoten) eine Verbesserung der Approximationsgüte erreichen zu wollen.

Ein Schritt in Richtung Eindeutigkeit der Rekonstruktion könnte hinsichtlich einer Randbedingung vorgenommen werden, die die Planheit des erhaltenen Bandes einbezieht. Mit abnehmender Banddicke wird es immer weniger möglich, das relative Profil (das Austrittsprofil bezogen auf die Austrittshöhe) zu verändern, ohne die Planheit zu zerstören (siehe auch [39] und die dortige Referenz). Wenn die Umkehrung gilt, so hat sich bei einem planen Band das (relative) Profil proportional oder als eine definierte Funktion der Banddicke verändert. Diese Bedingung ließe sich in die Kostenfunktion des Gradientenabstieges oder allgemein in eine Optimierungsfunktion einbinden. Erhalten wird ein Modell, bei dem die Einhaltung dieser Bedingung von vornherein definiert ist und welches nur beim Vorliegen planer Bänder angewendet werden darf.

5.3.3 Hybridsystem aus linearem und neuronalem Prädiktor

Eine modellfreie Rekonstruktion der Zwischenprofile hat sich also als nicht möglich herausgestellt. Um eindeutige Zwischenprofilverläufe zu berechnen, **muss** ein formales Gerüstmodell mit eingebracht werden. Der Ansatz mit reinen neuronalen Netzen hat bestätigt, dass Zwischenprofile unterschiedlichen Verlaufes generiert werden, je nach Netztyp und Initialisierung. Da das vorhandene analytische Modell bekannte Probleme bei der Zwischenprofilbestimmung hat, die im Detail in der Sensitivitätsanalyse aufzeigt worden sind und andererseits gezeigt werden kann, dass eine lineare Modellierung der gesamten Walzstraße ähnlich gut zur Endprofilvorhersage geeignet ist, versuchten wir über eine lineare Modellierung des Einzelgerüste auch Zwischenprofile zu erhalten.

Dabei werden alle Gerüste durch die gleiche lineare Abbildung beschrieben, d.h. das auslaufende Profil als Linearkombination von einlaufendem Profil und den gerüstspezifischen Parametern dargestellt,

$$p_n = \alpha p_{n-1} + a_1 W_n + a_2 B + a_3 D_n + a_4 \tag{7}$$

Allgemein geschrieben für beliebig viele Parameter P_i, $P_1 = D, P_2 = B, P_3 = W$

$$p_n = \alpha p_{n-1} + \sum_i a_i P_{i,n} \tag{8}$$

Die Koeffizienten α und a_i werden für alle Gerüste als identisch angenommen. N-faches Hintereinanderschalten dieser Gerüstübertragungsfunktion ergibt dann die Transferfunktion für das Profil über die gesamte Walzstraße

$$p_N = \alpha(\ldots\alpha(\alpha p_1 + \sum_i a_i P_{i,1}) + \sum_i a_i P_{i,2})\ldots) + \sum_i a_i P_{i,N} \tag{9}$$

Zusammengefasst und nach den a_i sortiert:

$$p_N = \alpha^N p_0 + \sum_i \left(a_i \sum_{n=1}^{N} \alpha^{N-n} P_n \right) \tag{10}$$

Dabei bleibt diese Funktion linear in den Koeffizienten a_i. Der Koeffizient α hingegen taucht in unterschiedlichen Potenzen auf. Er kann als eine Art Decay-Faktor angesehen werden, der angibt wie stark der Einfluss weiter vorn in der Walzstraße liegender Gerüste auf das Endprofil abnimmt.

Dieses Modell auf einen Datensatz anzupassen bedeutet, Werte für die Koeffizienten α und a_i zu finden, die den Fehler des Endprofils minimieren. Bei gegebenem α können die optimalen a_i durch lineare Regression z.B. mittels der Pseudoinversen schnell bestimmt werden. Das optimale α muss durch Näherungsverfahren bestimmt werden, da es als Lösung eines Polynoms N-ten Grades für $N > 4$ nicht analytisch berechnet werden kann. Da es aber der einzige freie Parameter ist und das Intervall, in dem es liegen kann, eingeschränkt ist, können Standardverfahren zur Minimierung wie Newton eingesetzt werden. Sinnvolle Lösungen für α müssen zwischen 0 (Endprofil hängt nur vom letzten Gerüst ab) und 1 liegen.

Test auf den Walzdaten

In Abb. 51 ist die Approximationsgüte des linearen Modells in Abhängigkeit von α dargestellt. Man sieht, dass ein klares Minimum vorhanden ist. Optimiert und getestet wurde auf dem gesamten Datensatz. Als optimales α ergab sich ein Wert von 0.64, d.h. das einlaufende Profil geht mit diesem Faktor in das Austrittsprofil ein. Im Vergleich zum analytischen Modell ist die Genauigkeit der Endprofilvorhersage um 5% schlechter. Grund kann die fehlende Online-Adaption dieses Modells sein. Auf den Umstellungsbändern hingegen liefert die lineare Modellierung eine gegenüber dem analytischen Modell um 20% genauere Endprofilvorhersage, siehe Tabelle 27. Auf den Umstellungsbändern werden dessen Adaptionskoeffizienten wieder auf Standardwerte gesetzt.

Allerdings ist auch das lineare Modell auf den Umstellungsbändern deutlich schlechter, das ist nur aus abweichenden Güten der Daten erklärbar, da ja keine Adaption stattfindet. Wahrscheinlich ist das berechnete Walzspaltprofil auf den Umstellungsbändern nicht korrekt.

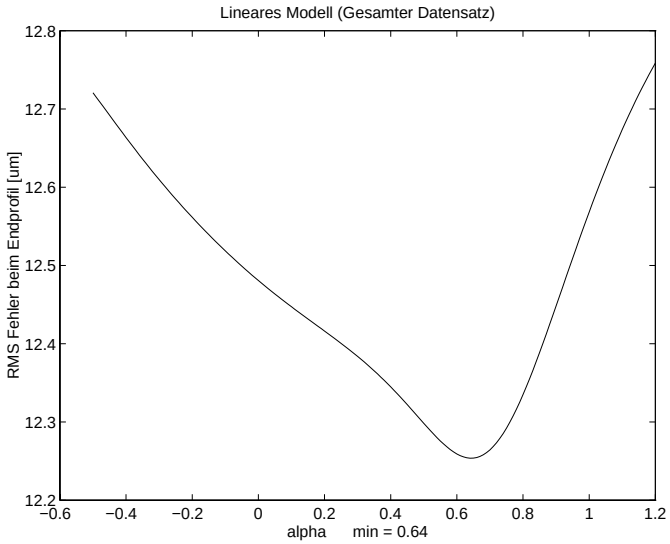


ABBILDUNG 51: Einfluss des “Decay-Faktors” α auf die Genauigkeit der Endprofilvorhersage. Ein eindeutiges Minimum liegt bei 0.64.

TABELLE 27: Güte der Approximation mit dem linearen Modell, angegeben ist der RMS Fehler

Methode	**Alle Bänder [μm]**	**Umstellungsbänder [μm]**
Mittelwert	13.3	17.5
Analytisches Modell	11.8	18.7
Gerüstweise lineares Modell	12.3	15.5
Gerüstweise lineares Modell (optimiert mit quadratisch linearer Approximation)	11.3	13.7

Der Verlauf der Zwischenprofile unterscheidet sich deutlich von dem des analytischen Modells, der Verlauf ist regelmäßiger. Jedes nichtlineare Modell bringt z.T. extrem unterschiedliche Verläufe hervor, d.h. Die Wahl des korrekten Modells ist kritisch für die Zwischenprofile. Der lineare Ansatz besitzt hohe Stabilität bei den geringsten willkürlichen Annahmen.

Kombination aus linearem und neuronalen Prädiktor

Das iterierte lineare Modell liefert eindeutige Zwischenprofile, ist aber nicht in der Lage, die Physik des Walzprozesses so gut zu approximieren wie es mit einem besseren Modell möglich wäre: Das Endprofil lässt sich mit deutlich höherer Genauigkeit vorhersagen, z.B. mit einem neuronalen Netz, das auch die Nichtlinearitäten modellieren kann. Eine Kombination beider Methoden liefert die Möglichkeit, die hohe Präzision der modellfreien Endprofilvorhersage mit der an ein Modell gebundenen Zwischenprofilberechnung zu verbinden.Dazu haben wir entsprechend Abbildung 52 folgendes Verfahren entwickelt:

1. Auswahl eines Datensets zur Bestimmung der Parameter des linearen Gerüstmodells, z.B. durch ein gleitendes Fenster oder besonders repräsentative Datensätze. Eventuell kann das Online-Endprofilmodell die Auswahl vornehmen.

2. Vorhersage des Endprofils des aktuellen Bandes mit diesem Modell
3. Vorhersage des Endprofils des aktuellen Bandes mit dem besten vorhandenen Modell zur Endprofilbestimmung
4. Vergleich der Ergebnisse und Korrektur der Koeffizienten des linearen Gerüstmodells, so dass das gleiche Endprofil herauskommt. Dabei sollen die Koeffizienten so wenig wie möglich geändert werden. Sie gelten nur für das aktuelle Band.

Die Änderung der Koeffizienten sollte so erfolgen, dass die Summe aller Änderungen von a_i und α so klein wie möglich ist. Damit bleibt das lineare Gerüstmodell trotz der Korrektur dem ursprünglichen optimal angepassten Modell so ähnlich wie möglich. Das ist auf zwei Weisen erreichbar: Bei normierten Daten sollte die Änderung aller Koeffizienten gleich groß sein, d.h. der Euklidische Abstand zwischen altem und neuen Koeffizientenvektor minimal sein. Als Normierungsverfahren sollten dann alle Eingänge so skaliert werden, dass alle Koeffizienten des linearen Modells zu eins werden. Bei nichtnormierten Eingängen sollten alle Koeffizienten mit einem möglichst ähnlichen Faktor multipliziert werden.

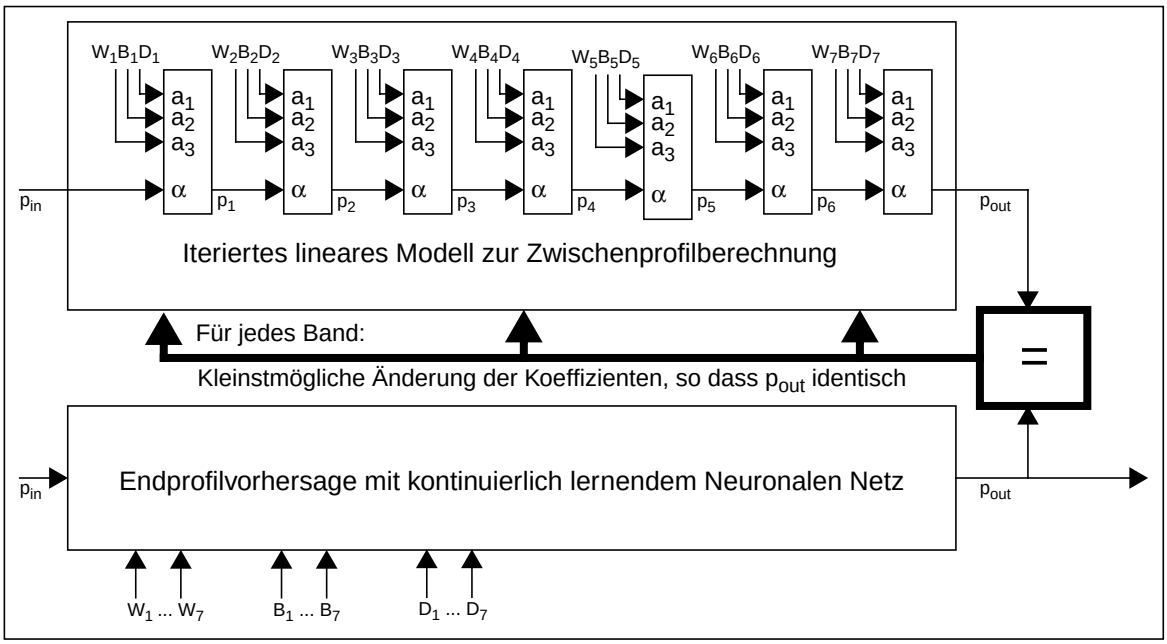


ABBILDUNG 52: Hybrides Modell zur Profilvorhersage bei einer 7-gerüstigen Walzstraße. Das Gerüstmodell berechnet aus einlaufendem Profil p_{in} und den Gerüsteinstellungen X,Y, Z die Zwischenprofile p_n und das Endprofil p_{out}. Das Neuronale Netz berechnet ebenfalls das Endprofil p_{out}, allerdings in der Regel genauer. Daher werden die Parameter a,b,c des Gerüstmodells so adjustiert, dass es ein möglichst ähnliches Endprofil wie das NN erzeugt.

Die Anwendbarkeit einer solchen Kombination ist nicht auf ein rein lineares Modell zur Zwischenprofilvorhersage beschränkt. Auch für andere, die Physik des Walzprozesses besser beschreibende mathematische Gleichungen könnte man deren freie Parameter zunächst global optimieren und dann für jedes Band so modifizieren, dass sich das vom einem anderen Modell prognostizierte Endprofil ergibt.

Abb. 53 zeigt zum Vergleich die Profilverläufe von analytischem Modell und Hybridmodell, jeweils nur auf den Umstellungsbändern; in Tabelle 27 ist die Güte der Approximation angegeben.. Zum Training wurden alle vorhandenen Bänder vor dem aktuellen und alle ab 1000 Bändern nach dem aktuellen Band verwendet. Datensatz: 12020 Bänder, 806 Umstellungsbänder.

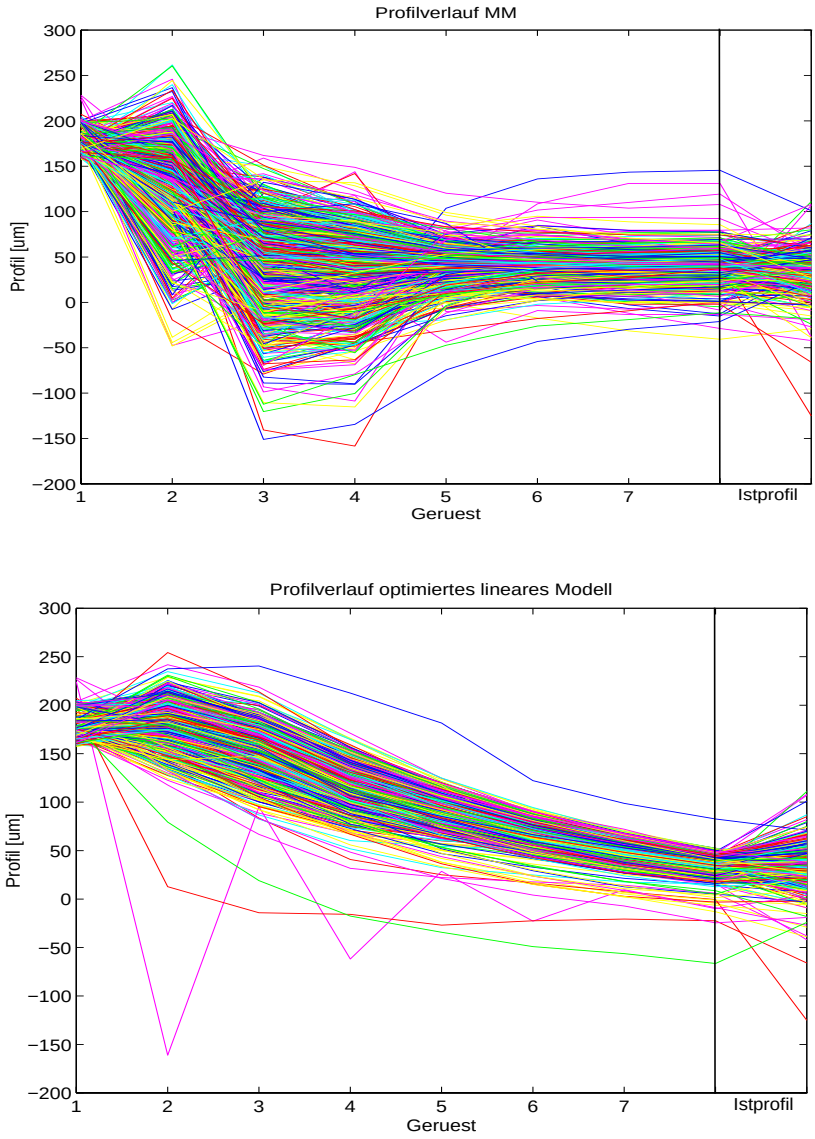


ABBILDUNG 53: Vergleich des Zwischenprofilverlaufes auf 806 Umstellungsbändern im analytischen und im linearen Modell. Ganz links das Eintrittsprofil, dann die (berechneten) 6 Zwischenprofile und Endprofil und zum Vergleich rechts noch das gemessene Endprofil.

5.3.4 Ergebnisse

1. Das vorhandene analytische Modell gewinnt seine Leistungsfähigkeit hauptsächlich aus der Adaption. Eine Linearisierung in den Arbeitswalzendurchmessern führt zu einer geringfügigen Verbesserung der Vorhersage des Endprofils.

2. Ein modellfreier Ansatz zur Zwischenprofilbestimmung ist nicht möglich. Es existieren

unendlich viele Transferfunktionen - identisch für jedes Gerüst -, die zum gleichen Endprofil, aber zu unterschiedlichen Zwischenprofilverläufen führen. Zur eindeutigen Bestimmung der Zwischenprofile ist ein mathematisches Modell des Walzprozesses zwingend notwendig; die erhaltenen Profile sind gültig unter dem verwendeten Modell und der verwendeten Optimierungsstrategie.

3. In Ermangelung eines korrekten Modells ist die lineare Approximation die “einfachste” Annahme. Sie liefert Endprofile mit einer dem bisherigen Modell vergleichbaren Genauigkeit, bei Umstellungsbändern ist sie sogar exakter. Die berechneten Zwischenprofilverläufe weisen einen etwas anderen Verlauf auf als bisher.
4. Eine modellfreie Systemmodellierung (direkte Abbildung ohne Iteration) für die gesamte Walzstraße mit einem neuronalen Online-Verfahren führt zu deutlich besseren Endprofilprognosen. Die Kombination beider Verfahren liefert somit ein Modell, das hohe Endprofilvorhersagegenauigkeit mit eindeutigen Zwischenprofilverläufen liefert.
5. Eine Evaluierung des Zwischenprofilverlaufes ist nur durch Messung der tatsächlichen gewalzten Zwischenprofile möglich. Alternativ wäre eine Qualitätsverbesserung bei Einsatz der neuen Methode in der Stichplanung ein Indiz.

Diese Form der Parallelschaltung von Modellen hat insbesondere auch Vorteile bezüglich der Zuverlässigkeit und Sicherheit der Verfahren im industriellen Einsatz. Das mathematische Modell ist robust und kann bei Bedarf ohne die neuronale Optimierung verwendet werden, auch wenn dies zu suboptimalen Ergebnissen führt. Erst wenn das Netz durch ausreichendes Training verläßlich bessere Werte liefert, wird sein Ausgangswert zur Optimierung der Parameter des mathematischen Modells herangezogen.

6 Zusammenfassung

In diesem Bericht wurden die Arbeiten und Ergebnisse dargestellt, die am FORWISS im Rahmen des Verbundprojekts AENEAS im Zeitraum vom 1.10.1995 bis zum 31.12.1999 erzielt wurden. Die Forschungsziele des Vorhabens wurden durch eine industrielle Anwendung im Bereich der Stahlverarbeitung motiviert und konzentrierten sich im Wesentlichen auf die folgenden Punkte:

- Modellierung von nichtlinearen und zeitvarianten Prozessen, die analytisch nicht fassbar sind und nur durch Messdaten repräsentiert werden.
- Modellierung von Größen, die nicht direkt messbar sind, aber auf nichtlineare Weise von anderen, messbaren Größen abhängen.
- Kombination von analytischen bzw. statistischen Modellen und Neuronalen Netzen, um die jeweiligen Vorteile der Verfahren zu vereinen.

Als Ergebnis des Vorhabens wurden eine Reihe neuer Ansätze zum kontinuierlichen Lernen entwickelt, darunter eine neuartige, lebenslang adaptive Netzarchitektur mit entscheidenden Vorteilen im Bereich des kontinuierlichen Lernens im Vergleich zu allen bisher bekannten Verfahren.

Zum zweiten Punkt wurde eine Theorie der Analyse iterierter Prozesse entwickelt, die auf das mathematische Problem der Lösung von Funktionswurzeln führte. Für nichtlineare Systeme gibt es keine analytischen Lösungsmöglichkeiten, daher wurden erstmals Neuronale Netze zur Lösung dieses Problems verwendet.

Die Ergebnisse aller grundlagenorientierten Arbeiten flossen in die Lösung eines industriellen Anwendungsproblems ein, bei der End- und Zwischenprofile warmgewalzter Stahlbänder modelliert und prognostiziert werden sollten. Dieser Prozess ist charakterisiert durch Nichtlinearität, Zeitvarianz („Tagesform“ der Anlage) und durch die nicht direkte Messbarkeit der Zwischenprofile, die sich als inverse Iteration (Funktionswurzel) aus dem Endprofil ergeben. Dieses Problem konnte auf elegante Weise durch eine Verknüpfung von analytischen und neuronalen Ansätzen gelöst werden.

Neben dem unmittelbaren Wert der Ergebnisse bei der Lösung der beispielhaften Anwendung lassen sich die entwickelten Verfahren zum kontinuierlichen Lernen und zur Analyse iterierter Prozesse auf eine Vielzahl anderer Problemstellungen verallgemeinern und stellen eine gute Basis für weitere Forschungsarbeiten dar.

7 Literatur

7.1 Verwendete Literatur

[1] White, D. & Sofge, D. (Eds.): Handbook of Intelligent Control - Neural, Fuzzy, and Adaptive Approaches. Van Nostrand Reinhold, New York, 1992.

[2] McCloskey, M. & Cohen, N.: Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. The Psychology of Learning and Motivation, 24, pp. 109-165, 1989.

[3] Box G.E.P., Jenkins G.M.: Time series analysis, forecasting and control. Holden Day, San Francisco, 1970.

[4] Ljung, L., & Söderström, T.: Theory and practice of recursive identification. Cambridge, MIT Press, 1986.

[5] Schaal, S., & Atkeson, C.G.: Constructive Incremental Learning From Only Local Information. Neural Computation, 10,8, pp. 2047-2084, 1998.

[6] V. Tresp, S. Ahmad and R. Neumeier (1994), Training neural networks with deficient data, in J.D. Cowan, G. Tesauro and J.Alspector (cds), Advances in Neural Information Processing Systems 6, Morgan Kaufmann, San Mateo, California.

[7] M. Rosenblatt, (1956), Remarks on some Nonparametric Estimation of a Density Function, Ann Math. Statist., 27, 1956, pp. 823-825.

[8] E. Parzen, (1962), On Estimation of a Probability Density Function and Mode. Ann of Math. Statist., 33 1065-1076.

[9] D.J.C. MacKay (1995) Bayesian neural Networks and density networks, Nuclear Instruments and Methods in Physics Research, Section A 354(1): 73-80.

[10] Ormoneit D.: Probability Estimating Neural Networks, Shaker, 1998

[11] Geman S., Bienenstock E. and Doursat R.: Neural networks and the bias/variance dilemma, Neural Computation, Vol. 4, pp. 1-58, 1992.

[12] Rumelhart D. E, Hinton G. E. and Williams R. J.: Learning Internal Representations by Error Propagation. Parallel Distributed Processing, pp. 318-362., MIT Press, Cambridge, MA, 1986.

[13] Fahlman S. E.,: An Empirical Study of Learning Speed in Back-Propagation Networks. Pittsburgh, PA, 1988.

[14] Riedmiller M., Braun H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. International Conference on Neural Networks. San Francisco, CA. 1993.

[15] Cybenko, G.: approximation by superposition of a sigmoidal function. Mathematics of Control, Signal and Systems 2, 303-314, 1989.

[16] Hornik, K., Stinchcombe M. and White H.: Multilayer feedforward networks are universal approximators. Neural Networks, 2, pp. 359-366, 1989.

[17] Hornik, K.: Some new results on neural network approximation. Neural Networks 6, 1069-1072, 1993.

[18] Murray-Smith R. and JohansenT. A.: Local Learning in Local Model Networks, In: Multiple Model Approaches to Modelling and Control,Chapter 7, pp. 185-210, R. Murray-Smith and T. A. Johansen (Eds.), Taylor and Francis, London, 1997.

[19] Poggio, T., Girosi, F.: Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks. Science 247, pp. 978-982, 1990.

[20] Park J. und Sandberg I.: Universal approximation using radial-bais-function networks. Neural Computation, Vol. 3, pp. 246-257, 1991.

[21] S. E. Fahlman S.E., Lebiere C.: The cascade-correlation learning architecture, Advances in Neural Information Processing Systems 2. Proceedings of the 1989 Conference, pp. 524cp-532, Morgan Kaufmann, 1990.

[22] Mohraz K. and Protzel P.: FlexNet - A Flexible Neural Network Construction Algorithm, Michel Verleysen (Eds.), pp. 111-116, Proc. ESANN'96, European Symp. on Artificial Neural Networks, Bruges, Belgium, 1996.

[23] Hecht-Nielsen, R.: Counterpropagation networks. Proc. of the IEEE First International Conference on Neural Networks, Vol. 2, pp. 19-32, San Diego, CA 1987.

[24] Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69, 1982.

[25] Jacobs, R.A., Jordan, M.I., and Barto, A.G. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. Cognitive Science, 15, 219-250, 1991.

[26] Jacobs, R.A., Jordan, M.I., Nowlan, S.J., & Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation, 3, pp. 79-87, 1991.

[27] Jordan, M.I., and Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6, pp. 181-214, 1994.

[28] Reilly D. L., Cooper L. N. and Elbaum C.: A neural model for category learning. Biological Cybernetics, 45:35-41, 1982.

[29] Berthold, M., Diamond J.: Boosting the Performance of RBF Networks with Dynamic Decay Adjustment. Advances in Neural Information Processing Systems 7, Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[30] Kuczama M. Choczewski B. and Ger R.: Iterative Functional Equations, Cambridge University Press, Cambridge, 1990

[31] Bishop CM.,Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995

[32] Feigenbaum MJ. Universal Behavoir in Nonlinear Systems, Los Alamos Science, Los Alamos, 1978

[33] Lindhoff, D., Sörgel, G., Gramckow, O. & Klode, K.-D.: Erfahrungen beim Einsatz Neuronaler Netze in der Walzwerksautomatisierung. *Stahl und Eisen*, 114, Heft 4, S. 49-53+208, 1994.

[34] Martinetz, T., Gramckow, O., Protzel, P. & Sörgel, G.: Neuronale Netze zur Steuerung von Walzstraßen. atp - Automatisierungstechnische Praxis, 38, Heft 10, S. 28-42, 1996.

[35] Isermann, R.: Identifikation dynamischer Systeme 1. Berlin, Heidelberg, New York, Springer Verlag, S. 235-248, 1992.

[36] Rieckmann J. Berechnung von Bandprofil und Planheit beim Kaltwalzen auf Sechs-Walzen-Walzwerken. Dissertation TH Aachen, Umformtechnische Schriften, Band 21, Verlag Stahleisen mbH, Düsseldorf, 1989

[37] Jahn J. (Institut für angewandte Mathematik, Universität Erlangen–Nürnberg) Persönliche Kommunikation.

[38] Wernstedt J & Otto P. NeuroControl. Vorlesung an der Technischen Universität Ilmenau, 1994

[39] Beisemann G. Theoretische Untersuchung der mechanisch einstellbaren Bereiche für die Walzspaltform an unterschiedlichen Walzwerksbauarten. Dissertation TH Aachen, Umformtechnische Schriften, Band 7, Verlag Stahleisen mbH, Düsseldorf, 1987

[40] French,R.M.: Catastrophic Forgetting in Connectionist Networks. Trends in Cognitive Sciences, 3(4), 128-135, 1999.

[41] Aamodt A. and Plaza E.: AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1, pp. 39-59, 1994.

[42] Cybenko, G.: Continous Valued neural networks with two hidden layers are sufficient. Technical report. Department of computer science, Tufts University. Medford, MA, 1988.

[43] Ronco E. and Gawthrop P. J.: Modular Neural Networks: a state of the art, Tech. Report CSC-95026, Centre for System and Control Department of Mechanical Engineering,University of Glasgow, Glasgow, Scotland, May 1997.

[44] Martinez D.: OFFSET: une methode de construction incrementale de reseaux de neurones multicouche et son application a la conception d'un autopilote automobile, PhD thesis Universite Paul Sabatier, Toulouse, France, 1992.

[45] Ronco E. and Gawthrop P.J.: Incremental Linear Controllers Network. American Control Conference (ACC'97), 1997.

[46] Ronco E., Gawthrop P.J. and Matter Y.: Incremental Modular Controllers Network. International Conference on Intelligent and Cognitive Systems (ICICS'96), 1996.

[47] Gawthrop P.J. and Ronco E.: Local Model Networks and Self-Tuning Predictive Control. IEEE Mediterranean Symposium on New Directions in Control and Automation, 1996.

[48] Moody, J., & Darken, C.: Learning with localized receptive fields. In: Touretzky, D., Hinton, G., & Sejnowski, T. (Eds.), Proceedings of the 1988 Connectionist Summer School, pp.133-143. San Mateo, CA: Morgan Kaufmann, 1988.

[49] D.E. Rumelhart and J. E. McClelland. Parallel Distributed Processing, MIT Press, Cambridge Mass., 1986

[50] Berthold MR & Diamond J. Boosting the Performance of RBF Networks with Dynamic Decay Adjustment. In: Tesauro G, Touretzky DS, and Leen TK (Eds): Advances in Neural Information Processing Systems 7, MIT Press, Cambridge MA, pp. 521-528.

[51] Grace A. Optimization Toolbox For Use with MATLAB. User's Guide. The Math Works Inc. 1994

[52] Rumelhart DE, Hinton GE & Williams RJ. Learning Internal Representations by Error Propagation. In: Rumelhart DE, McClelland JL and the PDP Research Group (Eds.). Parallel Distributed Processing: explorations in the microstructure of cognition, Chapter 8, pp. 318-362 MIT Press, Cambridge (Mass.), 1986

[53] Schmid F. Profil bestimmen. Maschinenmarkt 98 (1992) 35, Vogl Verlag und Druck KG Würzburg

[54] Haidukov PI. On searching a function from a given iterate [Russian]. Uc. Zap Buriatsk. Ped Inst. 15 1958, pp. 3-28

[55] Babbage C. Examples of the solutions of equations. Cambridge, 1820.

[56] Baker IN. Zusammensetzung ganzer Funktionen Math. Zeitschr. 69, 1958, pp.121-163

[57] Linden A., SESAME - ein objekt und datenflussorientierter Simulator für Modelle der Neuroinformatik und angrenzender Gebiete, Infix, Sankt Augustin, 1995

7.2 Entstandene Veröffentlichungen

[58] Martinetz T., Protzel P., Gramckow O. and Sörgel G.: Neural network control for steel rolling mills. In B. Kappen and S. Giele, editors, Neural Networks: artificial intelligence and industrial application, Berlin, 1995. Springer-Verlag.

[59] Martinetz, T., Gramckow, O., Protzel, P.: Walzwerksteuerung mit Neuronalen Netzen, In: VDI Bericht 1184, Neuronale Netze - Anwendungen in der Automatisierungstechnik, VDI/VDE-GMA Tagung in Langen, 4. Mai 1995, S. 35-42.

[60] Martinetz, T., Gramckow, O., Protzel, P. and Sörgel, G.: Neuronale Netze zur Steuerung von Walzstraßen, atp - Automatisierungstechnische Praxis, 38, Heft 10, pp. 28-42, 1996.

[61] Protzel, P., Kindermann, L., Tagscherer, M., Lewandowski, A.: Adaptive Systemidentifikation mit Neuronalen Netzen zur Profilsteuerung in Walzwerken, Computational Intelligence: Neuronale Netze, Evolutionäre Algorithmen, Fuzzy Control im industriellen Einsatz, VDI Berichte 1381, VDI Verlag, Düsseldorf, 1998, pp. 347-359.

[62] Tagscherer, M., Protzel P.: Adaptive Input-Space Clustering for Continuous Learning Tasks, Proceedings in Artificial Intelligence - FNS '98, Munich, Germany, Mar 1998, pp. 352-358.

[63] Tagscherer, M.: ICE - an Incremental Hybrid System for Continuous Learning, Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN'98), Skövde, Schweden, Sep 1998, pp. 597-602.

[64] Kindermann, L.: An Addition to Backpropagation for Computing Functional Roots, Proceedings of the International ICSC/IFAC Symposium on Neural Computation (NC'98), Vienna, 1998, pp. 424-427.

[65] Tagscherer, M.: Continuous Learning with an Incremental Hybrid System, Proceedings of the International ICSC/IFAC Symposium on Neural Computation (NC'98), Vienna, 1998, pp. 597-602.

[66] Kindermann, L.: Computing Iterative Roots with Neural Networks, Proceedings of the Fifth International Conference on Neural Information Processing (ICONIP'98), Kitakyushu, Japan, 1998.

[67] Tagscherer, M., Protzel, P.: Simultaneous learning of time-variant functions and data set distributions, Accepted for Presentation at the Sixth International Workshop Fuzzy-Neuro Systems '99 (FNS'99), Leipzig, Germany, March 1999, pp. 145-154.

[68] Kindermann, L., Trappenberg, T. P.: Modeling time-varying processes by unfolding the time domain, Proceedings of the International Joint Conference on Neural Networks (IJCNN'99), Washington DC, July 1999.

[69] Tagscherer, M., Protzel, P.: Kontinuierliches Lernen mit Neuronalen Netzen, 9. Workshop Fuzzy Control ’99, Dortmund, Germany, November 1999, pp. 108-121.

[70] Kindermann, L., Lewandowski, A., Tagscherer, M., Protzel, P.: Computing Confidence Measures and Marking Unreliable Predictions by Estimating Input Data Densities with MLPs, Proceedings of the Sixth International Conference on Neural Information Processing (ICONIP’99), Perth, Australia, Nov 1999, pp. 91-94.

[71] Lewandowski, A., Tagscherer, M., Kindermann, L., Protzel, P.: Improving the Fit of Locally Weighted Regression Models, Proceedings of the Sixth International Conference on Neural Information Processing (ICONIP’99), Perth, Australia, Nov 1999,pp. 371-374.

[72] Tagscherer, M., Kindermann, L., Lewandowski, A., Protzel, P.: Overcome Neural Limitations for Real World Applications by providing Confidence Values for Network Prediction, Proceedings of the Sixth International Conference on Neural Information Processing (ICONIP’99), Perth, Australia, Nov 1999, pp. 520-525.

[73] Protzel, P., Kindermann, L., Tagscherer, M., Lewandowski, A.: Abschätzung der Vertrauenswürdigkeit von Neuronalen Netzprognosen bei der Prozessoptimierung, VDI Bericht 1626 zur Fachtagung der VDI/VDE Gesellschaft für Mess- und Automatisierungstechnik: Computational Intelligence, Baden-Baden, Mai 2000, pp. 335-339.