

Michael Obach

**Zuverlässigkeit der Prognosen von
hybriden Neuronalen Netzwerken
und ihre Visualisierung**

mit Anwendungen in der Limnologie

Dissertation

Fachbereich Mathematik/Informatik

Universität Kassel

**Zuverlässigkeit der Prognosen
von hybriden Neuronalen Netzwerken
und ihre Visualisierung**

mit Anwendungen in der Limnologie

Dissertation

zur

Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

im Fachbereich Mathematik/Informatik
der Universität Kassel

vorgelegt von

Michael Obach

aus Kassel

Kassel, den 15. April 2003

Als Dissertation vom
Fachbereich Mathematik/Informatik
der Universität Kassel
angenommen.

Erster Gutachter: Prof. Dr. Heinrich Werner
Zweiter Gutachter: Prof. Dr. Rüdiger Wagner

Vorwort und Danksagung

Mit meiner Dissertation hoffe ich, eine sowohl für Mathematiker, Informatiker und Anwender Neuronaler Netzwerke als auch für Ökologen, Wasserbauingenieure und alle anderen Leser gleichermaßen interessante, nützliche und verständliche Darstellung meiner aktuellen Forschungsergebnisse vorzulegen.

Bezüglich der Form des Dokumentes habe ich versucht, ein ansprechendes und durch viele Verweise und Verzeichnisse zweckmäßiges Ergebnis zu erzielen. In der elektronischen Fassung als PDF-Dokument sind Verweise in Form von Hyperlinks vorhanden, um das Lesen und Nachschlagen zu erleichtern. Für Hervorhebungen im Text wurden verschiedene Formate verwendet, z. B. Namen von Software wie *Visualrbfn*, Dateinamen wie `visualrbfn.oct`, Datentypen wie `integer`, Dateiinhalte, Optionen von Software und Programmparameter wie `Quiet`, Stellvertreter einer Programmoption wie *Modus*, wissenschaftliche Gattungs- und Artnamen von Pflanzen und Tieren wie *Tinodes rostocki*, Internetadressen wie `http://www.Michael-Obach.de` und wichtige Begriffe wie *Neuronale Netzwerke*. Fachbegriffe, deren englischsprachige Übersetzungen in Abschnitt 6.5 ab S. 126 und im Stichwortverzeichnis zu finden sind, werden durch ^(e) gekennzeichnet. Bei der Silbentrennung von Dateinamen, Programmquelltexten und Internet-Adressen wurden keine Trennstriche und bei ihrer Stellung am Satzende keine Punkte verwendet, wenn dies missverständlich gewesen wäre. Als Dezimaltrenner dienen — entgegen den sonst beachteten Regeln der aktuellen Rechtschreibung — Punkte.

Einige in dieser Arbeit genannte Softwarebezeichnungen sind gesetzlich geschützt, auch wenn an den entsprechenden Stellen nicht explizit darauf hingewiesen wird.

Herr Prof. Dr. Heinrich WERNER und Herr Prof. Dr. Rüdiger WAGNER waren die wissenschaftlichen Betreuer meiner Promotion. Herr Prof. Dr. Rüdiger WAGNER stellte die biotische und Herr Dr. Hans-Heinrich SCHMIDT die abiotische Datengrundlage für meine Dissertation zur Verfügung. Meteorologische Messwerte lieferte der Deutsche Wetterdienst. Mit Herrn Dipl. biol. Reimo LIESKE führte ich viele anregende Diskussionen. Herr Prof. Dr. Peter ZWICK stellte mir einen Arbeitsplatz sowie Arbeitsmaterial zur Verfügung und unterstützte mich und meine Arbeit. Mit den Institutsangehörigen der Limnologischen Fluss-Station Schlitz des Max-Planck-Instituts für Limnologie und den Mitgliedern der Forschungsgruppe Neuronale Netzwerke an der Universität Kassel verbrachte ich viele Stunden, die ich in angenehmer Erinnerung behalten werde. Das rege Interesse meiner Verwandten und Bekannten bezüglich der Entwicklung meiner Dissertation empfand ich als motivierend. Die Arbeit wurde zeitweise durch die Max-Planck-Gesellschaft finanziell unterstützt. Einige Vorarbeiten förderte die Deutsche Forschungsgemeinschaft (BO 1012/5-3). Bei allen genannten Personen und Institutionen möchte ich mich herzlichst bedanken.

Mein ganz besonderer Dank gilt meinen Eltern, meinem Bruder Carsten und meiner Freundin Jone für ihre stete Zuneigung, Unterstützung und Geduld.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Dissertation selbstständig und ohne unerlaubte Hilfe angefertigt und andere als in der Dissertation angegebene Hilfsmittel nicht benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht. Kein Teil dieser Arbeit ist in einem anderen Promotions- oder Habilitationsverfahren verwendet worden.

Melsungen-Günsterode, den 15. April 2003

Michael Obach

Zusammenfassung

Die Approximation nicht-linearer und multidimensionaler Regressionsfunktionen durch Radiale-Basisfunktionen-Netzwerke, deren verdeckte Neuronen von Selbstorganisierenden Karten (SOM) unüberwacht trainiert werden, liefert im Gegensatz zu vielen anderen Prognosesystemen für jede Vorhersage eine Abschätzung der zu erwartenden Zuverlässigkeit dieses hybriden Künstlichen Neuronalen Netzwerkes (KNN) in Abhängigkeit von der Eingabe sowie für ein- oder zweidimensionale SOM-Gitter grafische Veranschaulichungen der Daten und des Netzwerkverhaltens.

In dem durch eine SOM partitionierten Eingaberaum lassen sich für jede Klasse die Anzahlen der in ihr enthaltenen Eingabedaten, Mittelwerte und Dispersionsmaße der eingeschränkten empirischen Verteilung der Ausgabegröße sowie mittlere Prognosefehler angeben. Nach der Hybridisierung mit Validity-Index-Netzen (VIN) können auf der Basis von Kreuzvalidierungen lokale Fehler sowie Vertrauens- und Vorhersageintervalle geschätzt werden. Neben dem minimalen Abstand zu den Klassenprototypen und dem maximalen Funktionswert lokaler radialer Basisfunktionen erlauben das Extrapolationsmaß des VINs und Likelihoods von Parzen-Dichteschätzern, Netzeingaben als bekannt oder neuartig zu klassifizieren.

Aus dem anschaulichen, wenige Voraussetzungen und Daten erfordernden Ansatz resultieren prinzipielle Schwierigkeiten bei der Abbildung intrinsisch hochdimensionaler Eingaberäume auf zweidimensionale SOM-Gitter und eine mögliche suboptimale Prognosequalität aufgrund des unüberwachten Lernens sowie praktische Probleme bei der Optimierung der Bandbreiten radialer Basisfunktionen und der Anzahl von SOM-Zellen. Das entwickelte Softwarepaket zur Simulation des KNN-Hybrids und anderer Modelle erzeugt zahlreiche Grafiken, die Lern- und Testdaten, Residuen und Szenarien für partielle Netzausgabefunktionen gemeinsam mit den genannten lokalen Zuverlässigkeitsmaßen in Linien- und Streudiagrammen veranschaulichen und eine visuelle Inspektion der verdeckten Schicht als Graustufendarstellungen dieser Werte für die SOM-Klassen, deren Prototypen oder einzelne Netzeingaben sowie Sequenzen von Eingabemustern auf dem SOM-Gitter ermöglichen. Nach der Entfernung irrelevanter und redundanter Eingabegrößen lassen sich die verbleibenden Eingabemerkmale zur Verbesserung der Generalisierungsfähigkeit des KNNs durch eine weitere, vorgeschaltete Schicht von Neuronen durch ein Gradientenabstiegsverfahren automatisch reskalieren.

Die Anwendbarkeit des Modells und der Software demonstrieren exemplarisch die Vorhersagen des täglichen Abflusses eines kleinen, hessischen Baches in Abhängigkeit von Niederschlags- und Temperaturwerten sowie die Prognosen der jährlichen Abundanz von dort in einer Emergenzfalle gefangenen Eintags-, Stein- und Köcherfliegenarten für veränderliche Umweltbedingungen auf der Basis eines Langzeitdatensatzes.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation der Arbeit	1
1.2	Überblick	7
2	Grundlagen und Konzepte	9
2.1	Ausgewählte Typen Neuronaler Netzwerke	9
2.1.1	Einführung	9
2.1.2	Selbstorganisierende Karten (SOMs)	11
2.1.3	Radiale-Basisfunktionen-Netzwerke (RBFNs)	16
2.2	Bestehende Ansätze zur Abschätzung lokaler Zuverlässigkeit	20
2.2.1	Lernen von Fehlerbalken	20
2.2.2	Validity-Index-Netzwerk	20
2.2.3	Bayesische Netzwerke	22
2.2.4	Weitere Ansätze	22
2.3	Hybridisierung von SOMs und RBFNs	26
2.3.1	Hybride Neuronale Netzwerke	26
2.3.2	RBFsOM — Anwendung einer SOM zur Positionierung von RBF-Zentren und zur Visualisierung	26
2.3.3	Reskalierung von Prädiktoren	30
3	Simulationssoftware	35
3.1	Bedienung und Benutzerschnittstelle	35
3.1.1	Datenvorverarbeitung	35
3.1.2	Format von Musterdateien	37
3.1.3	Programmausführung	38
3.1.4	Programmooptionen und Parameterdateien	40
3.1.5	Modifizierbare Shell- und Octave-Skripten	50
3.1.6	Ausgaben	51
3.1.7	Laufzeit und weitere Hinweise	57
3.2	Technische Aspekte	59
3.2.1	Systemvoraussetzungen und Installation	59
3.2.2	Programmierung des Netzwerksimulators in Octave	61
3.2.3	Erweiterung der SOM_PAK-Software	62
3.3	Diskussion	64

4	Fallstudien aus der Limnologie	67
4.1	Prognose der Abundanz von Wasserinsekten	67
4.1.1	Einführung	67
4.1.2	Material und Methoden	68
4.1.3	Ergebnisse	70
4.1.4	Diskussion	76
4.2	Wetter-Abfluss-Beziehung für einen Bach und sein Einzugsgebiet	80
4.2.1	Einführung	80
4.2.2	Datengrundlage und Vorverarbeitung	81
4.2.3	Methoden und Software	82
4.2.4	Ergebnisse	82
4.2.5	Diskussion	106
5	Abschlussdiskussion	111
6	Anhang	121
6.1	Artnamen und Abundanzmuster der untersuchten Eintags-, Stein- und Köcherfliegen	121
6.2	Optionen der Simulationssoftware	123
6.3	Abkürzungen	124
6.4	Häufig verwendete Symbole	125
6.5	Auswahl englischer Fachbegriffe	126
6.6	Software und Bezugsquellen	128
6.7	Abbildungsverzeichnis	129
6.8	Tabellenverzeichnis	132
6.9	Quellenverzeichnis	133
6.10	Stichwortverzeichnis	143

1 Einleitung

1.1 Motivation der Arbeit

Jedes Prognosesystem — ob Wahrsager, Wetterfrosch, Börsen-Guru oder *Künstliches Neuronales Netzwerk* — muss sich daran messen lassen, wie oft und wie genau seine Vorhersagen zutreffen. Gewisse *Prognosen*^(e) sind unter bestimmten Gegebenheiten einfacher und mit einer höheren Genauigkeit abzugeben als unter anderen Bedingungen. Beispielsweise ist eine Wettervorhersage für Deutschland während eines stabilen Sommerhochs in der Regel zutreffender als eine für typisches „Aprilwetter“. Damit hängt die Verlässlichkeit der Vorhersagen nicht nur vom Prognosesystem, sondern auch von den Voraussetzungen ab. Gleichzeitig möchte man meistens möglichst anschaulich verstehen, wie es zu einer Prognose gekommen ist.

An ein adaptives System, das aufgrund von Beobachtungen oder Messungen Prognosen abgibt, sind u. a. folgende Fragen zu richten:

- ▷ Auf wie vielen ähnlichen Beobachtungen, Messergebnissen bzw. Erfahrungen basiert die Prognose?
- ▷ Wie ähnlich ist die aktuelle Sachlage zu den Beobachtungen? Ist sie vergleichbar mit bekannten Gegebenheiten oder neuartig?
- ▷ Wie widersprüchlich waren die Messergebnisse der vorherzusagenden Größen in ähnlichen Situationen, war diesbezüglich eine große oder kleine Streuung vorhanden, weichen die unter ähnlichen Bedingungen erhobenen Daten stark voneinander ab?
- ▷ Wie nahe liegt der prognostizierte Wert an der bestmöglichen Vorhersage?
- ▷ Wie gut beschreibt das zugrunde liegende Modell die Messungen und Beobachtungen?
- ▷ Wie kann man die Prognosen und ihre Entstehung veranschaulichen und plausibel machen?

Eine Präzisierung und Eingrenzung der Problematik erfolgt unter Zuhilfenahme von Begriffen aus der Statistik und der Neuroinformatik, wobei wegen der leichteren Verständlichkeit und besseren Veranschaulichungsmöglichkeit zunächst exemplarisch nur zwei Merkmale betrachtet werden.

Gegeben seien *Daten* einer Stichprobe $((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$ mit reellen Zahlen x_i und y_i ($i = 1, \dots, n$) als Ergebnisse von Messungen der beiden metrisch skalierten Merkmale X und Y an n Objekten oder zu n Zeitpunkten. Besteht zwischen den Zufallsvariablen X und Y eine stochastische Abhängigkeit, lässt sich die Frage stellen, welchen Wert man für Y — im Mittel — unter der Bedingung erwarten kann, dass durch x ein Wert für X gegeben ist.

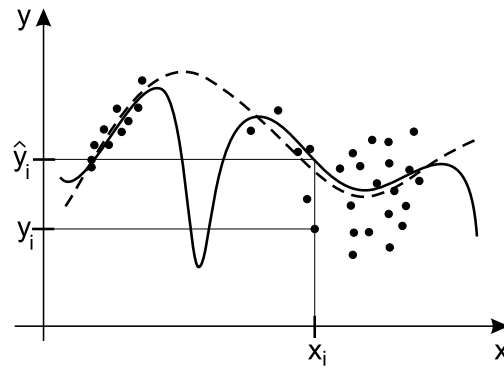


Abbildung 1.1: Ein unterstellter, aber unbekannter Zusammenhang zwischen zwei Größen X und Y mit einer Regressionsfunktion erster Art [GKN78, S. 479] (gestrichelte Kurve) wird näherungsweise durch ein Modell (durchgezogene, dicke Kurve) beschrieben, das an gegebene Datenpunkte angepasst wurde. Die Abweichungen zwischen den Prognosen $\hat{y}(x_i)$ und den Messwerten y_i an den Stellen x_i spielen bei der Beurteilung der Güte des Modells eine wichtige Rolle.

Bei solchen *Regressions-* oder *Kalibrationsproblemen* nennt man Y *abhängige* oder *Prognose-Variable* [Bor93, S. 167]. X wird als *unabhängige Variable* oder *Prädiktor* bezeichnet [BEPW96, S. 5].¹⁾

Der Zusammenhang werde näherungsweise durch eine an die vorliegenden, möglichst repräsentativen Daten angepasste Funktion $\hat{y}: \mathbb{R} \rightarrow \mathbb{R}$ modelliert, vgl. Abb. 1.1. Die bekannteste Approximationsfunktion stellt das lineare Regressionsmodell

$$\hat{y}(x) = \hat{w}_1 x + \hat{w}_0 \quad (1.1)$$

dar, wobei die Parameter \hat{w}_0 und \hat{w}_1 zu schätzen sind.

Im Hinblick auf KNNs heißt ein Wert x auch *Netzeingabe*^(e), der Raum, dem x entstammt, *Eingabe(daten)raum*, und die *Netzausgabe*^(e) $\hat{y}(x)$ liegt gemeinsam mit den Werten y_i der *Zielgröße*^(e) Y im *Ausgabe(daten)raum*.

Allgemein wird das Modell \hat{y} so gewählt und seine freien Parameter werden z. B. mit numerischen Verfahren so optimiert, dass die (vertikalen) Abweichungen oder *Residualgrößen* als Beträge der *Einzelfehler* oder *Residuen*²⁾ $\varepsilon_i := y_i - \hat{y}_i$ der Prognosen $\hat{y}_i := \hat{y}(x_i)$

¹⁾ Weitere Bezeichnungen für Y sind *Regressand* [Voß00, S. 177], *endogene* oder *erklärte Variable* sowie *Kriteriumsvariable* [BEPW96, S. 5], während X auch *Regressor* [Voß00, S. 177] sowie *erklärende* oder *exogene Variable* [BEPW96, S. 5] genannt wird.

²⁾ Zum Begriff *Residuum* vgl. [BEPW96, S. 13].

und der Messwerte y_i im Mittel möglichst klein werden. Für KNN-Modelle spricht man bei der Parameteranpassung vom *Lernen* oder *Training*. Eine zu minimierende Fehlerfunktion ist die Summe der quadrierten Abweichungen

$$E_{\text{SQE}} = \sum_{i=1}^n \varepsilon_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (1.2)$$

die auch *Fehlerquadratsumme*^(e) oder *Quadratsummenfehler* genannt wird.

Um die Modellgüte unabhängig von dem Stichprobenumfang und in den Einheiten der abhängigen Variablen abschätzen zu können, bietet sich die *Wurzel der mittleren Fehlerquadratsumme*, kurz RMSE, an:

$$E_{\text{RMS}} = \sqrt{\frac{1}{n} E_{\text{SQE}}} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}(x_i))^2} \quad (1.3)$$

(modifiziert nach [Bis95, S. 11]).

Als ein weiteres wichtiges Gütemaß, das unabhängig von der Varianz der Prognosevariablen und ihrer Skalierung ist, ist das (nicht-lineare) *Bestimmtheitsmaß*^(e)

$$B = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}(x_i))^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (1.4)$$

zu nennen.³⁾

Bei der Anpassung des Modells an die Daten besteht vor allem bei nicht-linearen Modellen die Gefahr, die Modellfunktion zu genau an die Daten der dafür verwendeten Stichprobe — im Falle von KNNs sogenannte *Lerndaten* — anzupassen, sodass sich die gewünschte Beschreibung des generellen Zusammenhanges in der Grundgesamtheit verschlechtern kann. Man spricht dann vom Problem der *Überanpassung*,⁴⁾ vgl. Abb. 2.7 (S. 24). Die *Generalisierungsfähigkeit* eines Modells versucht man abzuschätzen, indem die Gütemaße basierend auf einer nicht zur Anpassung des Modells verwendeten, unabhängigen und möglichst repräsentativen Stichprobe, sog. *Validierungs-* oder *Testdaten*, berechnet werden. Neben dem einfachen Aufteilen der Daten in Lern- und Testdaten (Split-Sample- oder Hold-Out-Validierung) verwendet man vor allem bei kleinen Stichprobenumfängen eine *Kreuzvalidierung* [Bis95, S. 375]. Dabei werden die Daten der zur Verfügung stehenden Stichprobe in S verschiedene, annähernd gleich große und oft zufällig ausgewählte Gruppen eingeteilt. Anschließend wird das zu testende Modell an die

³⁾ Stimmen alle durch \hat{y} vorhergesagten Punkte mit den gemessenen überein, verschwindet der Quadratsummenfehler und damit auch der Zähler des Bruches in Gl. 1.4. Dann ist B gleich eins, sonst ist B immer kleiner. Es sei vorausgesetzt, dass $n > 1$ und nicht alle Werte y_i für $i = 1, \dots, n$ gleich sind, da sonst der Nenner des Bruches verschwindet.

⁴⁾ Im Extremfall werden die Lerndaten interpoliert, was gelegentlich als „auswendig lernen“ bezeichnet wird. Dann steht normalerweise ein sehr kleiner Fehler bezüglich der Lerndaten einem vergleichsweise großen Fehler bei nicht zur Modellanpassung verwendeten Daten gegenüber.

Daten aus $S - 1$ dieser Gruppen angepasst und mit der zurückgehaltenen Teilmenge getestet. Nacheinander wird jede der S Gruppen als Test-Menge und der Rest als Trainingsgrundlage verwendet. Schließlich berechnet sich ein mittlerer Generalisierungsfehler als arithmetisches Mittel aus allen beobachteten Fehlern auf den verschiedenen Testdaten [Oba98, S. 56].⁵⁾

Die genannten Maßzahlen können zwar die Güte eines Modells als Ganzes näherungsweise beschreiben, allerdings sagen sie wenig über den — bezüglich der gewählten Metrik — *lokalen Fehler* aus, den man bei einer Prognose für einen gegebenen Wert x_0 oder (bei stetigen Funktionen) näherungsweise in einer Umgebung um ihn erwarten kann. In vie-

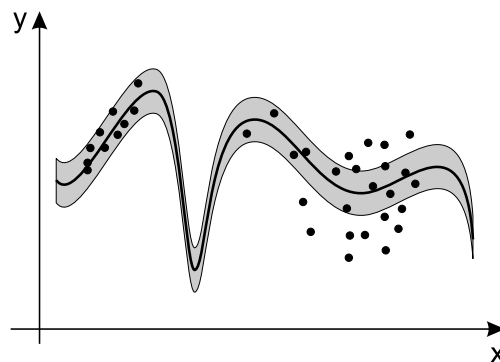


Abbildung 1.2: *Punktwolke mit Regressionsmodell und fälschlicherweise als konstant angenommenem Fehler. Die Annahme einer konstanten Streuung der Messwerte y_i um die Prognosefunktionswerte \hat{y}_i ist in vielen Fällen unzutreffend.*

len Fällen ist die Vorhersagegenauigkeit von \hat{y} nämlich nicht konstant, wie dies in Abb. 1.2 angedeutet ist, sondern sie variiert mit x . Ein Ziel eines Prognosesystems sollte es daher sein, zu einer Prognose auch die jeweils erwartete Genauigkeit etwa in Form des zu erwartenden Fehlers der Prognose $\hat{y}(x)$ in Abhängigkeit von x anzugeben, vgl. Abb. 1.3. Bei solchen Bereichs- oder Intervallschätzungen unterscheidet man einerseits zwischen *Vertrauens-* oder *Konfidenzintervallen*^(e) für $\hat{y}(x_0)$ an einer Stelle x_0 , die einen Bereich für die Lage des durch das Modell vorhergesagten Wertes (z. B. der bedingte Erwartungswert) schätzen, vgl. [Sar02] und [Voß00, S. 407 ff.], und andererseits geben *Vorhersage-* oder *Prognoseintervalle*^(e) für y_0 an, wo sich die Werte der Verteilung bei gegebenem x_0 mit einer gewissen Wahrscheinlichkeit befinden [Sar02, Voß00, S. 405 f.]. Vorhersageintervalle schließen Konfidenzintervalle ein [Hes97].

Abb. 1.3 verdeutlicht zudem, dass in Bereichen des Eingabedatenraumes, in denen wenige oder keine Daten vorliegen und in dem das KNN vor dem Problem einer *Extrapolation* [Sar02] steht, weder die Prognose noch die Abschätzung des zu erwartenden Fehlers zuverlässig sein können.

⁵⁾ Der Spezialfall, bei dem der Testdatensatz jeweils aus nur einem Datenpunkt besteht, wird *Leave-one-out-Methode* genannt.

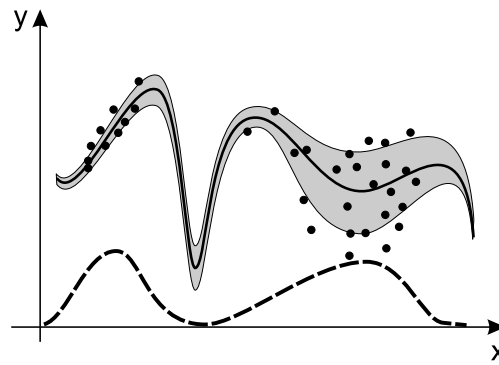


Abbildung 1.3: Zur Abschätzung der lokal zu erwartenden Güte eines an die gegebenen Daten (Punkte) angepassten Modells (dicke, durchgezogene Kurve) gehören Schätzungen zu den von x abhängigen Fehlern (grau unterlegte Fläche) und der Dichte der Punkte x_i (gestrichelte Kurve).

Man braucht daher ein Maß für den Grad der Extrapolation oder für die Dichte der Datenpunkte im Eingabedatenraum, um Eingabemuster als neuartig zu erkennen, bei denen Vorhersagen nicht durch ähnliche Beobachtungen gestützt sind.

In der vorliegenden Arbeit werden die Möglichkeiten untersucht, die sich durch eine als *Hybridisierung* bezeichnete Verknüpfung verschiedener KNNs ergeben, wobei eine Netzwerk-Komponente des KNN-Hybrids eine Partitionierung des Eingabedatenraumes in m Klassen \mathcal{X}_k durchführt, die dann jeweils durch einen *Prototypen* z_k repräsentiert sind, vgl. Abb. 1.4

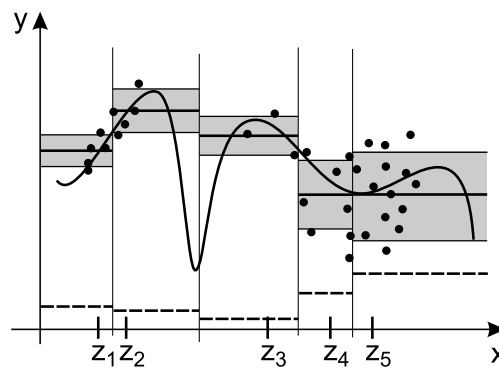


Abbildung 1.4: Der Eingabedatenraum, der in diesem Fall die Menge aller Punkte auf der Abszisse (x -Achse) enthält, kann durch m Vektoren z_k zerlegt werden, indem jeder Punkt x dem nächst-liegenden z_k (ggf. zusätzlich mit dem kleinsten Index) zugeordnet wird. Für jede Partition lassen sich u. a. Lageparameter wie der Median oder der Mittelwert, Streuungsmaße wie Quantile oder die Varianz (durch die Standardabweichung eingeschlossene Bereiche sind durch graue bzw. gerasterte Flächen angedeutet) sowie die Anzahl der in ihr liegenden Datenpunkte (gestrichelte Linien) angeben.

Durch die Partitionen der Eingabedaten x_i zerfällt auch die Menge der zugehörigen y_i in Klassen, für deren Elemente sich Maße der zentralen Tendenz und Dispersionsmaße wie z. B. der Median, das arithmetische Mittel, verschiedene Quantile und die Standardabweichung berechnen lassen. Die Mittelwerte \bar{y}_k der in einer Klasse liegenden y_i bilden eine einfache Schätzung für die Regression und somit einen Vergleichswert zu anderen Modellen.

Die Anzahl der in einer Klasse \mathcal{X}_k liegenden Eingabedaten x_i liefert einen Hinweis darauf, auf wie vielen ähnlichen Beobachtungen eine Prognose $\hat{y}(x)$ ($x \in \mathcal{X}_k$) basiert. Sind die Prototypen innerhalb der Datenpunktwolken positioniert, kann man den maximalen Abstand von x zu allen z_k bestimmen und wenn dieser eine vorher festgelegte Entfernung überschreitet, von einer unbekanntem Eingabe sprechen, vgl. [Tra01, S. 38]. Benutzt man — wie in Abb. 1.5 angedeutet — zum Aufbau des Modells eine additive, gewichtete Überlagerung (Superposition) von lokalen Funktionen $\phi_k(x)$, die in einiger Entfernung eines Zentrums z_k zumindest näherungsweise verschwinden, lässt sich auch mit diesen ein einfacher Extrapolationsdetektor konstruieren. Liegt an einer Stelle x_0 der maximale Funktionswert aller dieser Funktionen nämlich unter einer vorzugebenden Schranke, kann x_0 als nicht ausreichend bekannt und somit als neuartig eingestuft werden.

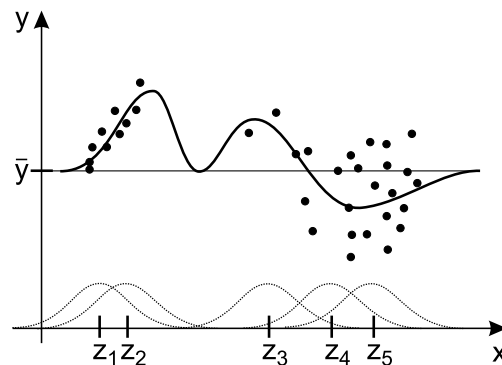


Abbildung 1.5: Durch die Superposition von „glockenförmigen“ radialen Basisfunktionen (gepunktete Kurven), die jeweils an einem Punkt z_k ($k = 1, \dots, 5$) zentriert sind, kann der Zusammenhang zwischen X und Y approximativ beschrieben werden. In Bereichen des Eingabedatenraumes (siehe x -Achse) mit wenigen Daten lässt sich die Funktion z. B. dem Mittelwert \bar{y} annähern.

In dem bisher betrachteten Fall mit zwei Zufallsgrößen X und Y und ihren reellen Ausprägungen x und y ist die Veranschaulichung der Sachverhalte verhältnismäßig einfach. Im Folgenden sei jedoch auch ein mehrdimensionaler Eingabedatenraum zugelassen und es gelte $\mathbf{x} \in \mathbb{R}^d$ mit $d \geq 1$. Das bedeutet, dass eine Prognose $\hat{y}(\mathbf{x})$ einer — auch weiterhin als eindimensional vorausgesetzten — abhängigen Variablen Y auch auf *Mustern*^(e) $\mathbf{x} = (x_1, \dots, x_d)$ als Realisierungen von *Zufallsvektoren* \mathbf{X} mit mehreren gemessenen Merkmalen X_j ($j = 1, \dots, d$) basieren kann. In diesem Fall sind Visualisierungstechniken zur Beurteilung der Güte der Modelle von Nutzen, die im Rahmen dieser Arbeit auf der Basis von KNNs entwickelt werden. Der Ansatz verwendet nachbarschaftserhaltende

Abbildungen der Daten aus einem höher dimensionalen Eingabedatenraum auf ein ein- oder zwei-dimensionales Gitter, auf dem sich die genannten und viele weitere Maße zur Einschätzung der lokalen Zuverlässigkeit von Netzwerkprognosen anschaulich darstellen lassen und die gleichzeitig eine Gruppierung der Daten liefern.

Mit der vorliegenden Dissertation wurden folgende Ziele bezüglich der Theorie und der Anwendungen von KNNs verfolgt:

- ▷ Anwendung und Weiterentwicklung bestehender Ansätze zur Abschätzung lokaler Zuverlässigkeit von KNNs
- ▷ Darstellung und Entwicklung von Möglichkeiten der Analyse und visuellen Inspektion der sonst verdeckten Elemente eines KNNs als Beitrag zur Veranschaulichung seiner Arbeitsweise
- ▷ Verbesserung der Prognosequalität eines bekannten KNN-Typs durch Reskalierung der Prädiktoren
- ▷ Entwicklung und Dokumentation einer Software zur Computersimulation der Modelle und Methoden, die sich für viele Anwendungssituationen eignet, leicht anwendbar ist und viele Optionen bietet sowie Informationen in gängigen Text- und Grafikformaten speichert
- ▷ Illustration und Überprüfung der Anwendbarkeit des Modells und der Software in Fallstudien
- ▷ Diskussion der neuen Ansätze, ihrer Umsetzung in Software und der darauf basierenden Anwendungen

Darüber hinaus sollen exemplarisch auf grundlegende und wichtige Fragestellungen aus der Fließgewässerforschung und der Ökologie anwendbare Modelle erstellt werden, die neben Prognosen Abschätzungen über ihre zu erwartende lokale Zuverlässigkeit liefern.

1.2 Überblick

Nach der allgemeinen Einleitung und der Erläuterung der Problemstellung finden sich im zweiten Kapitel zunächst Grundlagen zu den im weiteren Verlauf der Arbeit behandelten Typen von KNNs, anschließend aus der Fachliteratur bekannte Ansätze zur Abschätzung der Extrapolation, der Dichte und der Prognose- und Konfidenzintervalle und schließlich die in dieser Arbeit vorgestellten Konzepte. Die im Rahmen dieser Dissertation entwickelte Software zur Computersimulation des vorgestellten Netzwerktyps wird bezüglich der Bedienung und technischer Aspekte im dritten Kapitel erläutert. Im vierten Kapitel folgen Resultate und Diskussionen für zwei Fallstudien, in denen die Modelle und Verfahren auf ausgewählte Problemstellungen der Limnologie angewendet wurden. Eine allgemeine Diskussion der Methoden und Ergebnisse schließt den Hauptteil der Arbeit ab.

2 Grundlagen und Konzepte

2.1 Ausgewählte Typen Neuronaler Netzwerke

2.1.1 Einführung

Die ersten einfachen mathematischen Modelle von tierischen bzw. menschlichen Nervenzellen existieren schon seit dem Anfang der 1940er Jahre. Ein enormer Aufschwung bei der Anwendung von Simulationen *Künstlicher Neuronaler Netzwerke* (KNN) war aber erst im Zuge der fortschreitenden Computertechnologie in den 1980er Jahren und noch stärker im vergangenen Jahrzehnt zu beobachten.¹⁾ Zahlreiche Typen von KNNs werden aufgrund ihrer allgemeinen Merkmale — dazu zählen Lernfähigkeit, Robustheit, Fehler-toleranz, Generalisierungsfähigkeit und Parallelverarbeitung [Sch97, S. 5] — in außerordentlich vielfältigen Anwendungsgebieten eingesetzt.²⁾

Die Aufgabenbereiche von KNNs bei der Datenanalyse und Modellierung liegen neben der in dieser Arbeit im Vordergrund stehenden Regressionsschätzung, die sich auf Zeitreihen anwenden lässt, auch in der Gruppierung von Objekten, der Klassifikation und der Dimensionsreduktion. Damit ähneln die Anwendungen von KNNs denen statistischer Modelle und Verfahren wie der Regressions-, der Zeitreihen-, der Cluster-, der Diskriminanz- und der Faktorenanalyse.

Grundlegender Baustein eines KNNs ist das *Neuron*, entsprechend einer Nervenzelle im natürlichen Nervensystem. Es vermittelt mit p Eingängen und einem Ausgang eine Funktion $g: \mathbb{R}^p \rightarrow \mathbb{R}$ und besitzt freie Parameter, die als *Gewichte* oder *Synapsenstärken* bezeichnet werden. Der Ausgabewert heißt im Folgenden *Aktivität* oder *Erregung* des Neurons.

¹⁾ Zur Geschichte der KNNs vgl. [Zel94, S. 28 ff.] sowie [Sch97, S. 7 ff.]. Zum Aufbau und zur Funktionsweise von Nervenzellen aus Sicht der Biologie (Neurophysiologie) siehe z. B. [Sch97, S. 33 ff.], [Zel94, S. 35 ff.] oder eine ausführlichere Darstellung in [Ewe90].

²⁾ Eine Auswahl von Anwendungen der KNNs in Stichworten, siehe u. a. [Zel94, Sar02, Sch93, Sch97, Klö94]: Modellierung von biologischen Nervensystemen für Biologie und Psychologie, Erweiterung statistischer Methoden in der Datenanalyse in vielen wissenschaftlichen Disziplinen (für den Einsatz in der Limnologie werden in dem Kapitel 4 (ab S. 67) Beispiele vorgestellt sowie weitere Literatur angegeben), industrieller Einsatz in der Robotersteuerung, Erschaffung künstlicher Ohren und Nasen für Motorgeräuschtests und Weinproben, Qualitätskontrolle von Lacken und Fließbetonplatten, Text-, Sprach- und Gesichter-Erkennung, Aktienkursprognosen, Datenkompression, Rauschunterdrückung in der Fernmeldetechnik, Freund-Feind-Erkennung im militärischen Bereich, Kunden-Clusterung, Kreditwürdigkeitsüberprüfung, Sonardaten-Interpretation zur Minensuche, Autopiloten für Fahrzeuge, Wetterprognosen, Prothesen-Steuerung, Erkennen von Krebsgeschwüren aus Röntgenaufnahmen, klassifizierende Auswertung von Elektroenzephalogrammen.

Zu einem Netzwerk zusammenschaltete Neuronen oder *Zellen* können in *Schichten*^(e) angeordnet sein, in denen die Informationsverarbeitung parallel abläuft, vgl. Abb. 2.1. Eingehende Signale gelangen in der Arbeitsphase des KNNs zunächst in die *Eingabe-*

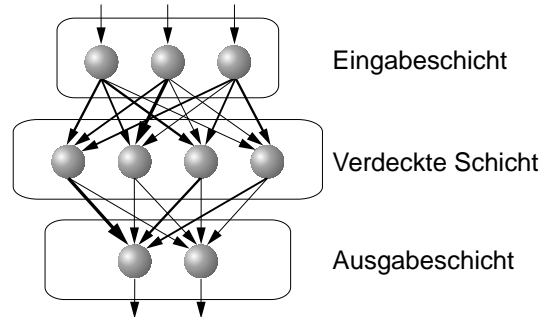


Abbildung 2.1: Schematischer Aufbau eines einfachen, vorwärts gerichteten Schichtennetzwerkes. Informationen werden entlang der Pfeile von einem Neuron zu einem anderen in der nächsten Schicht übertragen. Die Dicke der Pfeile gibt die Gewichtung der übertragenen Signale an.

schicht^(e), durchlaufen eine oder mehrere *verdeckte Schichten*^(e) und werden über eine *Ausgabeschicht*^(e) ausgegeben. *Vorwärts gerichtete Netze*^(e) haben keinerlei Rückkopplungen und vermitteln eine — in der Regel nicht-lineare und mehrdimensionale — Abbildung, siehe [Bis95, S. 120].³⁾ Beispiele dafür sind *Multi-Layer-Perzepton-Netze* (MLP), *Lineare Neuronale Netzwerke* (LNN) und die in Abschnitt 2.1.3 (ab S. 16) erläuterten *Radiale-Basisfunktionen-Netzwerke*.

Zwei wichtige Arten, die Gewichte der Neuronen beim Vorgang des *Lernens*^(e) oder *Trainierens* zu verändern, sind das überwachte und das unüberwachte Lernen. Beim *überwachten Lernen* kennt ein externer Lehrer — daher wird es auch *Lernen mit Lehrer* genannt — zu jedem Eingabemuster der Trainingsdaten das zugehörige Ausgabemuster [Zel94, S. 93 ff.] und passt mit Hilfe der Neuronengewichte die Netzausgaben so an die gegebenen Daten an, dass eine Kostenfunktion — im Spezialfall die Fehlerquadratsumme — minimiert wird.⁴⁾ Bei *unüberwachten Lernverfahren*^(e), die auch als *selbstorganisiert* oder als *Lernen ohne Lehrer* bezeichnet werden, sind keine Soll-Netzausgaben vorgegeben und es kann daher keine Fehlerfunktion wie beim überwachten Lernen verwendet werden [Sch97, S. 93]. Ein unüberwacht lernendes KNN wird im folgenden Unterabschnitt beschrieben.

³⁾ Nummeriert man die Neuronen eines solchen vorwärts gerichteten Netzwerkes beginnend mit den Eingabeneuronen über die verdeckten Einheiten bis zu den Ausgabezellen, erhalten alle Neuronen des KNNs nur Verbindungen von Neuronen, die einen kleineren Index haben als sie selbst [Bis95, S. 120].

⁴⁾ Beim Lernen mit Lehrer spricht man auch von einer *festen* und beim unüberwachten Training von einer *freien Lernaufgabe* [NKK94]. Eine weitere Bezeichnung für das Lernen mit Lehrer ist *Lernen mit Unterweisung* [RMS90, S. 118].

2.1.2 Selbstorganisierende Karten (SOMs)

Eine grundlegende Aufgabe des Denkens im Allgemeinen und insbesondere der explorativen Datenanalyse besteht darin, verschiedene gegebene Objekte in Gruppen zusammenzufassen, sodass die Elemente innerhalb einer Gruppe möglichst ähnlich sind, die Elemente verschiedener Gruppen sich aber stark unterscheiden. In der Statistik verwendet man zu diesem Zweck sogenannte *Clusteranalyseverfahren*.

KOHONENS⁵⁾ *selbstorganisierende* oder *sensorische Karten*^(e) (SOM) bieten überdies zahlreiche Visualisierungsmöglichkeiten für die Nachbarschaftsverhältnisse der Daten von metrischen Variablen, indem sie diese auch aus hochdimensionalen Räumen nach Möglichkeit nachbarschaftserhaltend auf niedriger dimensionale *Gitter*^(e) abbilden, siehe [Koh82], [Zel94, S. 591], [RMS90, S. 313] und [Bis95, S. 188 f.].

In einer SOM ist jedem Neuron i als Knoten eines regulären Gitters A ein *Referenzvektor* \mathbf{z}_k , auch *Gewichts-*, *Synapsenstärken-* oder *Kodebuchvektor* sowie *Erregungszentrum* oder *Prototyp* genannt, im Eingabedatenraum \mathbb{R}^d zugeordnet. Im Folgenden werden bevorzugt zweidimensionale, rechteckige Gitter mit einer Breite von m_b und einer Höhe von m_h Neuronen ($m_b \times m_h$ -SOMs) und eindimensionale Anordnungen ($m \times 1$ -SOMs, sog. *Ketten* [Zel94, S. 181]) betrachtet.⁶⁾

Im Arbeitsmodus wird ein Datenvektor \mathbf{x} aus dem \mathbb{R}^d auf den Knoten mit dem Index $c = \kappa(\mathbf{x})$ abgebildet, zu dessen Referenzvektor \mathbf{z}_c er bezüglich der gewählten Metrik eine minimale Distanz aufweist und in diesem Sinne am ähnlichsten ist (*Nächster-Nachbar-Klassifizierung*);⁷⁾ im Falle gleichen Abstandes zu anderen Referenzvektoren sei vereinbart, dass das SOM-Neuron unter ihnen mit dem kleinsten Index „gewinnt“ und als *Sieger* hervorgeht.

Der Eingabedatenraum \mathbb{R}^d zerfällt durch die Kodebuchvektoren \mathbf{z}_k ($k = 1, \dots, m$) einer SOM in Klassen

$$\mathcal{X}_k := \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x} - \mathbf{z}_k\| \leq \|\mathbf{x} - \mathbf{z}_\ell\| \quad \forall \ell = 1, \dots, m; k \text{ minimal}\},$$

sodass jeder Eingabevektor genau einer Partition angehört. Die Klassen \mathcal{X}_k werden von nun an *Einzugsgebiet*, *rezeptives Feld* oder *Zuständigkeitsbereich* [RMS90, S. 162] eines

⁵⁾ Auch die nach dem finnischen Wissenschaftler Teuvo KOHONEN benannten SOMs sind biologisch motiviert. Für alle Sinne des Menschen (mit Ausnahme des Geruchssinns) konnte man topographische Abbildungen^(e) zwischen Zellen sensorischer Organe und Cortex-Neuronen im Gehirn nachweisen. In [RMS90, S. 85 ff.] wird Kohonens Modell am Beispiel der Projektion des Raumes der Ultraschallfrequenzen auf den auditiven Kortex bei Fledermäusen simuliert, vgl. auch [Sch97, S. 94 ff.].

⁶⁾ Das Gitter kann zwischen den einzelnen Knoten neben den in dieser Arbeit vorausgesetzten rechteckigen auch hexagonale Verbindungen haben [KHKL96, S. 4], über die Informationen ausgetauscht werden. Neben den genannten diskreten Gittern gibt es auch sogenannte selbstorganisierende Oberflächen (self-organizing surfaces) [Zel94, S. 506 ff.].

⁷⁾ In der Regel benutzt man die Metrik, die durch die *euklidische Norm* induziert wird:

$$a(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| = \sqrt{\sum_{j=1}^d (x_j - z_j)^2}.$$

SOM-Neurons k genannt. Sie bilden konvexe Mengen und liefern für den zweidimensionalen Eingaberaum \mathbb{R}^2 ein *Voronoi-Diagramm* [Roj93, S. 367 f.].⁸⁾ Abb. 2.2 veranschaulicht eine solche *Voronoi-Parkettierung*^(e) der Ebene.

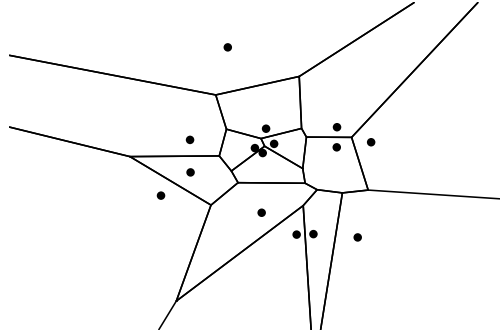


Abbildung 2.2: Beispiel einer Voronoi-Parkettierung der Ebene durch 15 zufällig ausgewählte Punkte (modifiziert nach [Wol99, S. 106]).

Jedem Vektor $\mathbf{x} \in \mathbb{R}^d$ lässt sich somit eine Klasse $\mathcal{X}_{\kappa(\mathbf{x})}$ als das Einzugsgebiet zuweisen, in dem er liegt. Die Menge der Lerndaten, die im Einzugsgebiet eines SOM-Neurons k liegen, wird im Folgenden durch \mathcal{X}_k^L und ihr Umfang durch $\#\mathcal{X}_k^L$ abgekürzt.

Während des *unüberwachten Lernens* der SOM aktivieren sich die Zellen des Gitters gegenseitig, die in einer bestimmten Umgebung des Gewinners liegen. Damit lernen diese dann ebenfalls von der Eingabe. Weiter entfernte Neuronen sollen hingegen davon nicht beeinflusst werden [KHKL96, S. 5]. Nach einem erfolgreichen Training sind benachbarten Knoten des SOM-Gitters zueinander ähnliche Prototypen-Vektoren im Eingabedatenraum zugeordnet. Sind m Neuronen im Gitter A und folglich m Gewichtsvektoren \mathbf{z}_k ($k = 1, \dots, m$) sowie n Trainingsdatenpunkte \mathbf{x}_i ($i = 1, \dots, n$) im \mathbb{R}^d gegeben, besteht das iterative Lernverfahren aus folgenden Schritten, vgl. [KHKL96, S. 5 f.], [BFM96, S. 63], [RMS90, S. 74] sowie [Sch97, S. 101]:

1. *Initialisierung:* Setze den Iterationszähler t auf null und starte mit geeigneten — falls keine weiteren Informationen bekannt sind, z. B. zufällig gewählten — verschiedenen Synapsenstärken $\mathbf{z}_k(t)$, kurz \mathbf{z}_k . Die Referenzvektoren werden dabei im Eingabedatenraum verteilt, siehe Abb. 2.3.
2. *Stimuluswahl:* Wähle gemäß einer Wahrscheinlichkeitsverteilung — im Spezialfall einen beliebigen oder bei dem zyklischen Präsentieren der Muster den nächsten — Vektor \mathbf{s} aus den Trainingsdaten $\{\mathbf{x}_i \mid i = 1, \dots, n\}$ im Eingaberaum.
3. *Antwort:* Bestimme das Siegerneuron mit dem *Erregungszentrum* $\mathbf{z}_c(t)$, vgl. Abb. 2.4.

⁸⁾ Die *Voronoi-Polygone* werden auch *Dirichlet-Regionen* oder *Thiessen-Polytope* genannt, siehe <http://mathworld.wolfram.com/VoronoiDiagram.html>.

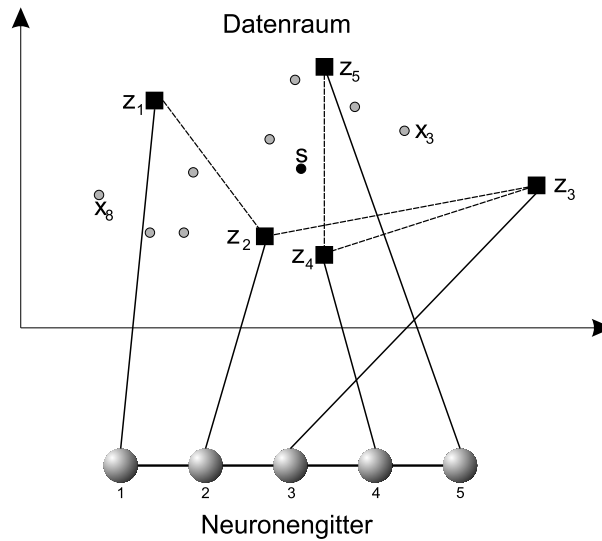


Abbildung 2.3: Veranschaulichung eines Neuronengitters A als eindimensionale, selbstorganisierende Karte (SOM). Ihre Referenz- oder Kodebuchvektoren \mathbf{z}_k ($k = 1, \dots, 5$) liegen im zweidimensionalen Eingaberaum \mathbb{R}^2 . Eine gestrichelte Linie verbindet die Darstellung der Referenzvektoren gemäß der Anordnung der zugehörigen Neuronen in A . Von den zum Training dienenden Datenpunkten \mathbf{x}_i ($i = 1, \dots, 8$) wird während des iterativen Lernvorgangs immer wieder ein beliebiger als „Stimulus“ \mathbf{s} ausgewählt.

4. *Adaptionsschritt:* Verändere alle Synapsenstärken für $k = 1, \dots, m$ gemäß

$$\mathbf{z}_k(t+1) = \mathbf{z}_k(t) + \eta(t) \cdot \psi_{ck}(t) \cdot (\mathbf{s} - \mathbf{z}_k(t)) . \quad (2.1)$$

Erhöhe den Zähler t um 1 und gehe zu Schritt 2, solange t kleiner als eine vorgegebene maximale Lernschrittzahl ist. Sonst beende das Verfahren.

Die genauso wie der nachfolgend verwendete *Nachbarschaftsradius* $r(t)$ mit fortschreitendem Lernvorgang immer kleiner werdende *Lernrate* $\eta(t)$ lässt sich z. B. als

$$\eta(t) = \frac{A}{B + t}$$

mit Konstanten A und B oder einfach linear wählen [KHKL96, S. 5].

Bezeichne a_{ck} den euklidischen Abstand der Neuronen c und k im Gitter A , wobei dieses im \mathbb{R}^2 eingebettet sei und der Abstand zwischen direkt benachbarten Gitterpunkten eins betrage. Eine oft verwendete *Nachbarschaftsfunktion*^(e) ist die *Gauß-Funktion*⁹⁾

$$\psi_{ck}(t) = \exp\left(-\frac{a_{ck}^2}{2r^2(t)}\right) .$$

⁹⁾ Bezeichne $\exp(x) = e^x$.

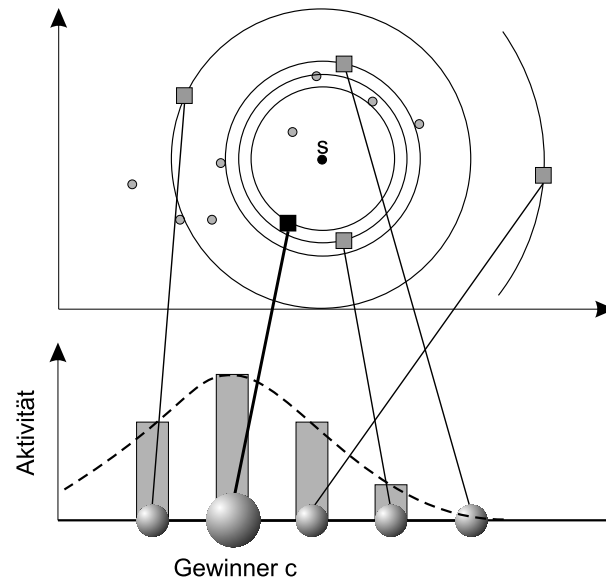


Abbildung 2.4: Der Abstand aller Referenzvektoren vom „Stimulus“ s (vgl. Abb. 2.3) wird ermittelt. Das Neuron c aus dem Neuronengitter A wird als Gewinner bezeichnet, für das dieser Abstand minimal ist (bei gleichem Abstand entscheidet der kleinste Index). Es hat, symbolisiert durch die Balkengrafik, maximale Aktivität, während die Erregungsstärke der anderen Neuronen des Gitters von dem (in A gemessenen) Abstand zu c und der Wahl der Nachbarschaftsfunktion ψ (gestrichelte Kurve) abhängt.

Eine Auswahl weiterer Nachbarschaftsfunktionen ist beispielsweise in [KHKL96, S. 5], [Sch97, S. 99], [Hof93, S. 117] und [Zel94, S. 181 f.] zu finden.

Der Referenzvektor $z_c(t)$ des Gewinnerneurons c nähert sich am stärksten dem Stimulus s . Die Gewichtsvektoren seiner Nachbarn verschieben sich — je nach Wahl des Radius und der Nachbarschaftsfunktion ψ — in Richtung zu s , während sich die Referenzvektoren von im Neuronengitter weit entfernt liegenden Elementen (wie in Abb. 2.5 illustriert) nicht oder nur sehr wenig verändern.¹⁰⁾ Die Erregungsstärke eines Neurons k und damit die Änderung eines Gewichtes ist nicht allein vom Abstand des Referenzvektors zum Stimulus im Eingabedatenraum \mathbb{R}^d abhängig, sondern vom Abstand des Neurons k vom Gewinnerneuron c im Gitter A .

Die Anzahl der für das Lernen nötigen Schritte kann 100'000 übersteigen [KHKL96, S. 7]. Dabei ist die Konvergenz dieses *Markov-Prozesses* nicht in jedem Fall gesichert, vgl. [RMS90, S. 76 und 265 ff.] sowie [Sch97, S. 102 ff.].

Gewöhnlich werden alle Eingabedaten als gleich wichtig betrachtet und dem Netz beim Lernen gleich häufig „gezeigt“. Sollen bestimmte Muster stärker repräsentiert sein, lassen sich diese (etwa mit der Software *SOM_PAK*) durch häufigeres Zeigen beim Training stär-

¹⁰⁾ Bei der „Mexikanerhut-Funktion“ $\psi_{ck}^M(t) = 3 \exp(-a_{ck}^2/(2r^2(t))) - 2 \exp(-a_{ck}^2/(4r^2(t)))$ [Hof93, S. 117] können sich einige Referenzvektoren auch von s weg bewegen.

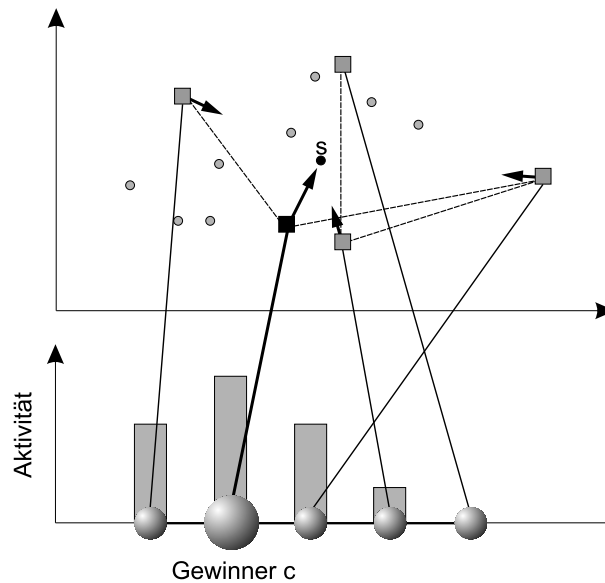


Abbildung 2.5: Die Position der Referenzvektoren im Datenraum V wird bei jedem Lernschritt in Abhängigkeit von der Aktivität der Neuronen zum „Stimulus“ s hin verändert, wie dies durch die Pfeile an den Darstellungen der zu verschiebenden Kodebuchvektoren angedeutet ist (vgl. Abbildungen 2.3 und 2.4).

ker gewichten [KHKL96, S. 6]. Allerdings können dadurch Dichteschätzungen verfälscht werden.

Ein Gütemaß für SOMs ist der über alle Datenpunkte x_i gebildete Mittelwert der *Quantisierungsfehler* $\|x_i - z_{\kappa(x_i)}\|$ [KHKL96, S. 6].

Neben der hier beschriebenen Variante einer SOM gibt es zahlreiche Modifikationen und Weiterentwicklungen. Beispiele für die Anwendung von SOMs in den Bereichen Zeichen-, Bild- und Spracherkennung, Telekommunikation, Prozesskontrolle, Robotersteuerung, Physik und Chemie sind in [Koh01, S. 358 ff.] gegeben.¹¹⁾ Weitere Literaturhinweise findet man u. a. bei [Zel94, S. 179 ff.], [RMS90], [Hof93, S. 116 ff.], [BFM96, S. 61 ff.] sowie [Sar02].

Es gibt zahlreiche Möglichkeiten der Visualisierung von Strukturen in Daten mit Hilfe von SOMs, von denen im Folgenden einige genannt sind, vgl. [KHKL96, S. 18 f.].

Die *Sammon-Abbildung*^(e) oder *Sammon-Projektion* ist eine nicht-lineare Projektionsmethode zur Abbildung von Vektoren eines höher dimensionalen Raumes in einen Raum niedrigerer Dimension [Sam69, Koh01, S. 35 ff.], wobei die relativen Abstände zwischen den Objekten möglichst erhalten bleiben [HDK00]. Werden die Gewichtsvektoren einer SOM durch eine Sammon-Abbildung in die Ebene projiziert, lassen sich die Bilder der Referenzvektoren von in dem SOM-Gitter direkt benachbarten Neuronen durch Linien verbinden. Dadurch erhält man für zweidimensionale, rechteckige SOMs netzartige Dar-

¹¹⁾ Über die Erkennung handgeschriebener Ziffern und die Gruppierung italienischer Olivenöle mit SOMs berichtet [Zel94, S. 409 ff.].

stellungen, wie sie in Abb. 4.2 (S. 70) und Abb. 4.9 (S. 85) zu sehen sind. Die Beschriftung der Repräsentation eines Kodebuchvektors \mathbf{z}_k ergibt sich aus der Bezeichnung des ersten Trainingsdatenpunktes, für den das SOM-Neuron k Sieger ist.

Über einer Darstellung eines ein- oder zweidimensionalen SOM-Gitters lassen sich einige Kennwerte der Daten bzw. der Kodebuchvektoren durch Graustufen der den SOM-Neuronen zugeordneten Grafikelemente veranschaulichen, wobei im Folgenden hohe Werte hellen und niedrige Werte dunklen Graustufen entsprechen. Insbesondere sind Werte von eins und darüber durch Weiß und solche von null und darunter durch Schwarz repräsentiert.

In einer *U-Matrix-Darstellung*¹²⁾ [Ult93] wird die relative Ähnlichkeit bzw. Unähnlichkeit der Neuronen zu ihren Nachbarn bezüglich des euklidischen Abstandes ihrer Gewichtsvektoren im Eingabedatenraum nicht nur durch ihre Position innerhalb der grafischen Repräsentation des SOM-Gitters, sondern auch durch die Graustufen der Füllung des Zwischenraumes zwischen je zwei Zellen wiedergegeben.¹³⁾ Dadurch sind Einblicke in die Strukturen der hochdimensionalen Daten möglich. Gruppen (Cluster) von ähnlichen Daten sind etwa durch helle Bereiche, die idealerweise durch dunklere Elemente getrennt sind, erkennbar, vgl. [Ult92]. Zwei Beispiele für mit der Simulationssoftware *SOM_PAK* erzeugte U-Matrix-Darstellungen finden sich in den Abbildungen 4.7 (a) auf S. 75 und 4.8 auf S. 84. Die Beschriftungen der einzelnen Repräsentanten der SOM-Neuronen entsprechen denjenigen der Sammon-Abbildung. Ist allerdings ein SOM-Neuron für keinen Trainingsdatenvektor Sieger, wird auf der ihm zugeordneten Zelle der U-Matrix ein gefüllter Kreis gezeichnet.

Abbildungen von Schnitten durch die verschiedenen Komponentenebenen^(e) zeigen, an welchen Stellen der SOM die jeweilige Komponente der Kodebuchvektoren hohe bzw. niedrige Werte aufweist, vgl. Abb. 4.10 (S. 86).

Für Zeitreihen ist es hilfreich, die Abfolge der Siegerneuronen für eine Sequenz von Eingabevektoren als Polygonzug entlang der entsprechenden Grafikelemente der SOM zu zeigen, die als *Trajektorien* bezeichnet werden, vgl. Abb. 4.13 (S. 87).

2.1.3 Radiale-Basisfunktionen-Netzwerke (RBFNs)

Ein *Radiale-Basisfunktionen-Netzwerk* [PG90], kurz RBFN, ist im einfachsten Fall ein vorwärts gerichtetes Schichtenetzwerk — siehe Abb. 2.1 (S. 10) — mit d Eingabeneuronen, welche die Komponenten eines d -dimensionalen Eingabevektors \mathbf{x} an alle m Neuronen mit Ausgaben $\phi_k(\mathbf{x})$ ($k = 1, \dots, m$) einer verdeckten Schicht leiten, und einem

¹²⁾ Der Begriff *U-Matrix* steht als Abkürzung für *Unified-Distance-Matrix*.

¹³⁾ Bei einem rechteckigen Aufbau der Verbindungen in der SOM sind die Graustufen der Füllungen der Kästchen zwischen zwei neben- oder übereinander angeordneten Neuronen erklärt. Für die Zuweisung der Graustufen von Füllungen von Zwischenräumen zwischen vier Neuronen ist die Summe der Abstände von Gewichtsvektoren jeweils diagonal gegenüberliegender Neuronen zu halbieren und entsprechend zu skalieren. Die Quadrate der Grafik, die den SOM-Neuronen selbst zugeordnet sind, werden mit der Graustufe gefüllt, die dem Mittel derjenigen der umgebenden Zwischenräume entspricht.

linearen Ausgabeneuron. Mit einem *Bias-Element* $\phi_0(\mathbf{x}) := 1$ ($\forall \mathbf{x} \in \mathbb{R}^d$) und den Gewichten $w_k \in \mathbb{R}$ ($k = 0, \dots, m$) der Ausgabezelle vermittelt ein solches RBFN nach [Bis95, S. 168] eine Funktion der Form

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^m w_k \phi_k(\mathbf{x}) + w_0 = \sum_{k=0}^m w_k \phi_k(\mathbf{x}). \quad (2.2)$$

Bei MLPs und LNNs hängt die Aktivität eines Neurons in einer verdeckten Schicht bzw. der Ausgabeschicht von einer gewichteten Summe seiner Eingaben sowie einem Schwellenwert ab. Im Gegensatz dazu ist die Erregung eines verdeckten RBFN-Neurons k durch den Wert einer Funktion ϕ_k des *Abstandes* $\|\mathbf{x} - \mathbf{z}_k\|$ einer Eingabe \mathbf{x} von einem zugehörigen *Zentrum* oder *Prototypen* $\mathbf{z}_k \in \mathbb{R}^d$ gegeben [Bis95, S. 182].¹⁴⁾ Der Abstand sei im Folgenden *euklidisch*, kann aber auch z. B. durch die City-Block-Metrik oder andere Distanzmaße bestimmt sein.

Die bekannteste *lokale radiale Basisfunktion*^(e) (RBF) oder *lokale Zentrumsfunktion* ist die *gaußsche Basisfunktion*^(e)

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}_k\|^2}{2h_k^2}\right) \quad (2.3)$$

[Bis95, S. 168].¹⁵⁾ Der Parameter h_k wird (*Band-)Breite*^(e) oder *Ausdehnung*^(e) genannt. Bei einer festen Anzahl m von Zentren und konstanten Bandbreiten $h_k = h$ für $k = 1, \dots, m$ hängt der Grad der Glättung der vermittelten Funktion \hat{y} von dem *Glättungsparameter* h ab. Sind für die einzelnen Basisfunktionen variable Bandbreiten h_k erlaubt, können diese beispielsweise als die Entfernung von den nächsten K Nachbarzentren gewählt werden, wodurch die Glattheit der Funktion \hat{y} mit K variiert.

Den Graphen eines RBFNs auf der Basis des euklidischen Abstandes und gaußscher Basisfunktionen, das für zudem konstante Bandbreiten im Folgenden als *gewöhnlich* bezeichnet wird, stelle man sich für skalare Eingaben als Überlagerung von „Glockenkurven“ vor, wie dies in Abb. 1.5 (S. 6) dargestellt ist.

Ein Beispiel für eine *nicht-lokale Basisfunktion* ist die *Thin-Plate-Spline-Funktion*

$$\phi_k(\mathbf{x}) = \|\mathbf{x} - \mathbf{z}_k\|^2 \ln(\|\mathbf{x} - \mathbf{z}_k\|) \quad (2.4)$$

[Bis95, S. 165 f.]. Weitere RBFs sind z. B. in [Bis95, S. 165 f.] und [Zel94, S. 229] aufgeführt.

Die Anzahl der Basisfunktionen wählt man oft wesentlich geringer als die Anzahl der Lerndaten. Dadurch wird die Komplexität des Netzwerkes derjenigen der näherungsweise zu beschreibenden Funktion und nicht dem Umfang des gegebenen Datensatzes nachempfunden. Ist für ein RBFN die Anzahl m der Zentren bestimmt, verläuft die Modellanpassung in zwei Stadien. Zunächst werden während des *Lernens* die Prototypen

¹⁴⁾ Die Punkte $\mathbf{z}_k \in \mathbb{R}^d$ werden auch als *Stützstellen* oder *Gewichte* bezeichnet [Zel94, S. 230].

¹⁵⁾ Zur Vereinfachung wird an dieser Stelle nicht zwischen Aktivierungs-, Transfer- und Ausgabefunktion eines Neurons unterschieden, vgl. [Hof93].

\mathbf{z}_k ($k = 1, \dots, m$) festgelegt, indem im einfachsten Fall zufällig oder willkürlich eine m -elementige Teilmenge der Trainingsdatenpunkte \mathbf{x}_i ($i = 1, \dots, n$) als Zentren selektiert wird. Es ist aber auch möglich, die Komponenten des Prototypenvektors zufällig aus dem jeweils beobachteten Wertebereich der einzelnen Prädiktoren zu wählen. Eine der elegantesten Möglichkeiten der Wahl der Zentren besteht in der Anwendung Daten-segmentierender Trainingsverfahren, die für jede der selbstorganisiert gefundenen Gruppen von Daten einen Prototypenvektor angeben. Beispiele dafür sind der K-Means-Algorithmus [Mac67], der als nicht-hierarchisches, partitionierendes Verfahren der *Clusteranalyse* bekannt ist [Bor93, S. 535 f.], oder die *Lernende Vektorquantisierung*^(e), kurz LVQ, siehe z. B. [Zel94, S. 239]. Auf die Anwendung von SOMs zum unüberwachten Positionieren der Zentren eines RBFNs [Sar02] wird in Abschnitt 2.3 ab S. 26 eingegangen. Anschließend erfolgt die Anpassung der Gewichte w_k ($k = 0, \dots, m$) des linearen Ausgabeneurons wie das Training eines LNNs¹⁶⁾ mit Methoden der numerischen Mathematik zur Lösung des linearen *Kleinstquadrat-Problems*^(e), wobei zur Berechnung einer sog. *Pseudo-Inversen* in der Praxis eine *Singulärwertzerlegung* [PTVF95, Mas93, S. 167] verwendet wird [Bis95, S. 92 und S. 171]. Darüber hinaus kommt auch ein iteratives und dadurch langsames, aber robustes und leicht zu programmierendes Gradientenabstiegsverfahren in Frage, siehe [Zel94, S. 236 ff.], [Sch97, S. 170] sowie Unterabschnitt 2.3.3 ab S. 30.

In einem weiteren Schritt lassen sich unter verschiedenen, angepassten Modellen geeignete auswählen, wodurch die Anzahl m der verdeckten radialen Neuronen und ihre Breiten h optimiert werden können. Damit nicht das nur bezüglich der Trainingsdaten am besten angepasste Netz als „optimal“ betrachtet wird, muss die Auswahl des besten Modells im Hinblick auf die approximative Beschreibung des Zusammenhanges zwischen \mathbf{X} und Y in der Grundgesamtheit auf einem unabhängigen Validierungsdatensatz erfolgen.

Es gibt viele Möglichkeiten der Modifikation von RBFNs. In der in dem Kapitel 3 dokumentierten Simulationssoftware wurde ein linearer Teil hinzugefügt, sodass das Netz die Funktion

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^m w_k \phi_k(\mathbf{x}) + \sum_{j=1}^d \hat{w}_j x_j + w_0 \quad (2.5)$$

vermittelt, vgl. [Zel94, S. 234]. Abb. 2.6 veranschaulicht die Superposition der einzelnen Komponenten.

Bei der Datenvorverarbeitung vor der Anwendung eines RBFNs mit einer linearen Ausgabeschicht ist eine Skalentransformation der abhängigen Variablen Y in ein (endliches)

¹⁶⁾ Die nur aus einer Ein- und einer Ausgabeschicht bestehenden LNNs vermitteln mit $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, $x_0 = 1$ und der Fehlerquadratsumme als Kostenfunktion eine multiple lineare Approximation

$$\hat{y}(\mathbf{x}) = \sum_{j=0}^d \hat{w}_j x_j$$

der Regressionsfunktion.

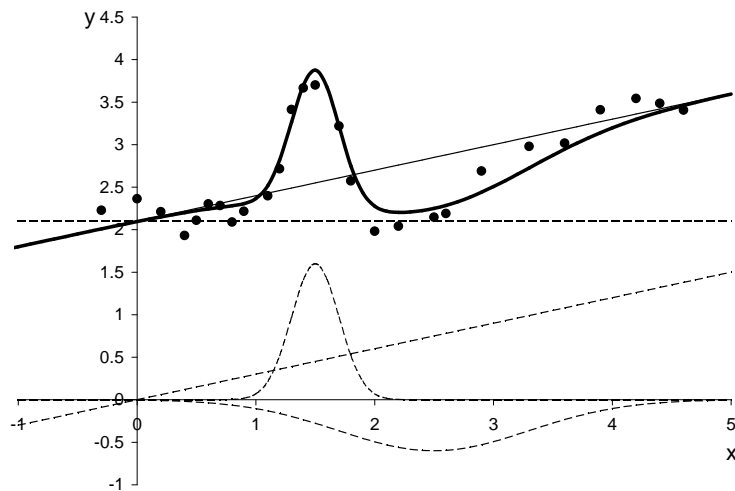


Abbildung 2.6: Veranschaulichung der Überlagerung (Superposition) einzelner Funktionen (die Ausschnitte ihrer Graphen sind durch gestrichelte Kurven dargestellt) zu einer Funktion \hat{y} . Hier setzt sich \hat{y} (der Graph ist als dicke Kurve gedruckt) aus $m = 2$ gewichteten radialen Basisfunktionen ϕ_k mit Zentren $z_1 = 1.5$ und $z_2 = 2.5$, Bandbreiten $h_1 = 0.2$ und $h_2 = 0.8$, Gewichten $w_1 = 1.6$ und $w_2 = -0.6$, einem konstanten Glied (Bias) $w_0 = 2.1$ sowie einem linearen Anteil mit der Steigung $w_1 = 0.3$ zusammen. Die Eingaberaum-Dimension ist $d = 1$. Der Graph der aus dem linearen Anteil und dem Bias-Element zusammengesetzten affinen Funktion (Asymptote), der sich \hat{y} für betragsmäßig große Werte von x nähert, ist als ein dünnes, „durchgezogenes“ Geradenstück illustriert.

Intervall, wie sie bei MLPs mit sigmoiden Ausgabeneuronen üblich ist, überflüssig. Wegen der radialen Funktionen in der verdeckten Schicht ist es ratsam, die Spannweiten oder die Varianzen der Eingabegrößen in ähnliche Größenordnungen zu transformieren, vgl. S. 36. Die Anwendung der *Nächste-Nachbarn-Heuristik* schwächt die Problematik gegenüber der Wahl geeigneter fester Bandbreiten etwas ab.

Sehr ähnlich zu RBFNs sind die nicht-parametrischen *Nadaraya-Watson-Kern-Regressionsschätzern* [Mic92, Bis95, S. 178] entsprechenden *General Regression Neural Networks* (GRNN) [Spe91, SH92, Oba98],

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^n y_i \cdot \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2h^2}\right)}{\sum_{i=1}^n \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2h^2}\right)} \quad (2.6)$$

[Bis95, S. 178].¹⁷⁾ Bei ihnen sind die hier als $\exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / (2h^2))$ gewählten *Kerne* auf allen Lern-Eingabedaten \mathbf{x}_i zentriert. Es gibt Modifikationen, bei denen weniger

¹⁷⁾ Der nach ihnen benannte Kern-Regressionsschätzer wurde von NADARAYA und WATSON unabhängig voneinander im Jahre 1964 angegeben [Mic92, S. 19]. Weitere Kerne sind in [Mic92, S. 16] zu finden.

Zentren verwendet werden [Spe91], was zu einem aufwendigeren und daher langsameren Training, aber zu einer höheren Ausführungsgeschwindigkeit führt.¹⁸⁾

2.2 Bestehende Ansätze zur Abschätzung lokaler Zuverlässigkeit

Eine Auswahl von Ansätzen zur Abschätzung lokaler, von den Netzwerkeingaben abhängiger Zuverlässigkeit einzelner Netzwerkeingaben bei Regressionsproblemen durch Dichteschätzungen, Extrapolationsdetektoren, Prognose- und Konfidenzintervalle findet sich in den folgenden Unterabschnitten.¹⁹⁾

2.2.1 Lernen von Fehlerbalken

Um die lokal zu erwartende Abweichung zwischen einer Prognose $\hat{y}(x_0)$ eines KNNs und dem vorherzusagenden, möglichen Messwert y_0 abschätzen zu können, lässt sich ein sehr einfaches Verfahren anwenden, das sog. *Predictive-Error-Bars* [LZ98] liefert. Dabei lernt ein weiteres KNN für jedes Element einer Menge von idealerweise bisher nicht für die Anpassung von \hat{y} verwendeten Daten die Fehlerquadrate $\varepsilon_i^2 = (y_i - \hat{y}_i)^2$ oder die absoluten Fehler $|\varepsilon_i| = |y_i - \hat{y}_i|$. Der bedingte Erwartungswert \hat{y} und die bedingte Varianz ε_i^2 lassen sich zugleich in einem einzigen Netzwerk schätzen [NW95].

Diese *Fehlerbalken*^(e) liefern allerdings keine Aussage über die Dichte der Trainingsdaten und daher keine Warnung bei einer Extrapolation. Außerdem sind neben den Trainingsdaten für das eigentliche Prognose-Netz noch weitere Daten nötig, um das zweite Netz für die Predictive-Error-Bars zu trainieren.

2.2.2 Validity-Index-Netzwerk

Das in [LKU92a, LKU92b] beschriebene *Validity-Index-Netzwerk* (VI-Netz) basiert auf einem modifizierten RBFN mit gaußschen Basisfunktionen und einem Bias-Element. Neben der Prognose $\hat{y}(x)$ liefert dieses KNN als weitere Ausgaben für jeden Eingabevektor x ein Extrapolationsmaß auf der Basis der geschätzten Dichte der Lern-Eingabedaten bei x sowie Schätzungen der Konfidenz- und Prognoseintervalle.²⁰⁾

Die Zentren z_k werden durch einen K-means-Algorithmus, die Breiten h_k danach durch eine Nächste-Nachbarn-Heuristik und schließlich die Gewichte w_k der Ausgabeschicht durch ein Verfahren zur Kleinstquadrat-Schätzung bestimmt, siehe. S. 18. Die Anzahl m

¹⁸⁾ Die Ergebnisse zahlreicher Anwendungen dieses Netztyps in ökologischen Fallstudien sind in [Oba98] nachzulesen.

¹⁹⁾ Die zahlreichen Ansätze zur Abschätzung der Konfidenz bei Klassifikationsproblemen bleiben in der vorliegenden Arbeit unberücksichtigt.

²⁰⁾ Für jede Ausgabedimension existiert im allgemeinen Fall ein weiteres Ausgabeneuron, das die Konfidenz- oder Prognoseintervalle liefert. Die Bewertung der Extrapolation ist skalarwertig, vgl. [LKU92b, S. 625].

der verdeckten Neuronen und die der nächsten Nachbarn für die Wahl der Bandbreiten lassen sich mit Hilfe von Kreuzvalidierungen bewerten.

Die Dichte wird mit n Trainingsdaten und den in Gl. 2.3 (S. 17) beschriebenen Basisfunktionen ϕ_k geschätzt:

$$\varrho(\mathbf{x}) = \frac{\sum_{k=1}^m \phi_k(\mathbf{x}) \varrho_k}{\sum_{k=1}^m \phi_k(\mathbf{x}) + 1 - \max \phi_k}. \quad (2.7)$$

Dabei ist²¹⁾

$$\varrho_k = \frac{n_k/n}{(\sqrt{2\pi} h_k)^d} \quad (2.8)$$

und

$$n_k = \sum_{i=1}^n \phi_k(\mathbf{x}_i).$$

Unter der Annahme, dass die Residuen unabhängig und mit verschwindendem Mittelwert normalverteilt sind sowie innerhalb der effektiven Reichweite einer jeden RBF eine annähernd konstante Varianz aufweisen, lassen sich die Konfidenzgrenzen für einen Eingabevektor \mathbf{x} durch

$$\gamma(\mathbf{x}) = \frac{\sum_{k=1}^m \phi_k(\mathbf{x}) \gamma_k}{\sum_{k=1}^m \phi_k(\mathbf{x})} \quad (2.9)$$

schätzen. Die Zahl t_{95} ist der kritische Wert der t -Verteilung (95% bei $n_k - 1$ Freiheitsgraden).²²⁾ Die Beiträge der einzelnen RBF-Zentren sind

$$\gamma_k = t_{95} s_k / \sqrt{n_k}. \quad (2.10)$$

Um ein Prognoseintervall zu berechnen, kann man stattdessen

$$\tilde{\sigma}_k = t_{95} s_k \sqrt{1 + 1/n_k}$$

verwenden. Eine Schätzung für die Varianz des Modellresiduums im Einzugsbereich der verdeckten Einheit k ist durch

$$s_k^2 = \frac{\sum_{i=1}^n \phi_k(\mathbf{x}_i) E_i^2}{n_k - 1}$$

gegeben, wobei E_i der Kreuzvalidierungsfehler für den Trainingsvektor mit dem Index i ist [LKU92b].

²¹⁾ Wegen des Unterschiedes im Nenner des Bruches im Exponenten der e -Funktion bei der Definition der Basisfunktionen zwischen der vorliegenden Arbeit und der Darstellung in [LKU92b] wurde der Faktor $\sqrt{2}$ bei der Berechnung des Volumens im Nenner des Bruches in Gl. 2.8 eingefügt.

²²⁾ Die Werte der t -Verteilung entnehme man Computersoftware (z. B. *Octave*, *SPSS*) oder Tafeln der Statistik-Literatur, etwa [Bor93].

2.2.3 Bayesische Netzwerke

Bei dem bisher beschriebenen frequentistischen Ansatz oder Häufigkeitsansatz, der in der Regel auf dem Maximum-Likelihood-Schätzer basiert, versucht man, für ein Modell bezüglich der gegebenen Daten *einen* optimalen Parametervektor zu finden.

Bei der Anwendung von *bayesischen* oder *bayesianischen Netzen*^(e) hingegen beschreibt eine Wahrscheinlichkeitsdichtefunktion die Unsicherheit in die Auswahl der Gewichte des Netzwerkes [Sar02, Bis95, S. 42]. Ohne das Vorliegen einer Stichprobe wird das Vorwissen über einen zu beschreibenden Zusammenhang zunächst durch eine *a-priori-Wahrscheinlichkeitsdichte*^(e) der Netzwerkparameter modelliert. Nach dem Training lässt sich die zugehörige *a-posteriori-Wahrscheinlichkeitsdichte*^(e) über die Netzwerkgewichte ermitteln. Parameterwerte, die zu einer schlechten Anpassung an die Daten führen, erscheinen dann weniger plausibel als zuvor, während die Plausibilität oder *Likelihood*⁽²³⁾ für die Trainingsdaten gut beschreibende Gewichte zugenommen hat.

Mit Hilfe der a-posteriori-Wahrscheinlichkeitsdichte lässt sich für einen beliebigen Eingabevektor nicht nur die Netzwerkausgabe in Form eines einzelnen Wertes (bzw. Vektors) angeben, sondern es sind auch Aussagen über die Verteilung der Netzausgaben und damit den Grad der Plausibilität möglich, mit der diese erzeugt wurde [Bis95, S. 385 ff.].⁽²⁴⁾

Gegenüber Maximum-Likelihood-Schätzungen sind KNN-Prognosen nach dem bayesischen Ansatz weniger durch systematische Fehler behaftet, aber die Netze sind schwieriger in Computerprogramme zu implementieren [DR01].

2.2.4 Weitere Ansätze

Ein ähnlicher Ansatz zur Dichteschätzung in einem VI-Netzwerk, um die Unsicherheit in Netzwerkprognosen in Abhängigkeit vom Grad der Unbekanntheit bzw. Neuartigkeit^(e) der Eingabevektoren zu beschreiben, besteht in nicht-parametrischen *Parzen-Dichteschätzern*. Das in [Bis94, S. 220] beschriebene Verfahren zur Dichteschätzung mit

$$\hat{p}(\mathbf{x}) = \frac{1}{n (\sqrt{2\pi} h)^d} \sum_{i=1}^n \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2 h^2}\right) \quad (2.11)$$

unterscheidet sich aber von dem Extrapolationsmaß des VI-Netzes u. a. dadurch, dass alle Trainingsdatenpunkte \mathbf{x}_i als Zentren von Gauß-Glocken und eine konstante Bandbreite h verwendet werden. Der Wert von h wurde in [Bis94] als das über alle Trainingsdaten gebildete Mittel der mittleren Abstände zu den jeweils 10 nächsten Nachbarn gewählt. Er lässt sich aber auch durch wiederholte Kreuzvalidierungen beurteilen und verbessern. Ein Eingabevektor wird als neuartig eingestuft, wenn der natürliche Logarithmus dieser

²³⁾ Der Begriff *Likelihood* hat sich auch in der deutschsprachigen Statistik-Literatur durchgesetzt, siehe etwa [Bor93, S. 96].

²⁴⁾ Zudem erlaubt die Verwendung bayesischer Techniken für KNNs die Auswahl einer geeigneten Netzwerk-Architektur (z. B. Anzahl der verdeckten Schichten oder die Anzahl von verdeckten Neuronen) sowie andere Anpassungen an die Daten, wie beispielsweise die Bestimmung einer geeigneten Glattheit der Funktion oder die Auswahl von relevanten Eingabevariablen, siehe [Mac92, Mac95, Nea93, Nea96].

Likelihood des Parzen-Schätzers unterhalb von etwa -5 liegt. Eine Weiterentwicklung mit variablen Bandbreiten zu semi-parametrischen Dichteschätzern findet sich in [Bis95, S. 62].

RBFNs lassen sich um ein weiteres Ausgabeneuron als *Extrapolationsdetektor* erweitern, das für eine Netzeingabe \mathbf{x} die Ausgabe

$$r(\mathbf{x}) = 1 - \max_{k=1, \dots, m} (\phi_k(\mathbf{x})) \quad (2.12)$$

liefert, wobei ϕ_k die Erregung des RBF-Neurons k darstellt [Loh01]. Bei lokalen (nicht-negativen) Aktivierungsfunktionen wird ein Muster \mathbf{x} umso weniger bekannt sein, je näher $r(\mathbf{x})$ an eins liegt, sodass Werte nahe eins Extrapolation andeuten. Die Aktivitäten von RBF-Neuronen mit lokalen RBFs wurden auch in [TKLP99] zur Erkennung von Bereichen des Eingaberaumes mit wenigen Daten verwendet. Zur Einschätzung der Dichte der Eingabedaten können auch *Wavelet-basierte Netze* dienen [SMZM97].

Ein Ansatz der Einschätzung lokaler Prognosegüte für Netze mit nicht-lokalen Aktivierungsfunktionen besteht darin, drei Netzwerke — beispielsweise MLPs, die gleich aufgebaut sein können — parallel mit den selben Eingabedaten aber unterschiedlichen Ausgaben zu verwenden [KLTP99, PKTL00]. Dabei lernt das erste Netz die Regressionsfunktion, das zweite den absoluten Fehler des ersten (siehe Unterabschnitt 2.2.1 auf S. 20) und das dritte dient zum Erkennen neuartiger Eingaben. Zum Training des Extrapolationsanzeigers wird allen Lern-Eingabedaten eine „Soll-Ausgabe“ von eins zugewiesen und für zufällig aus dem Eingabedatenraum gewählte Punkte wird der zu lernende Wert gleich null gesetzt. Allerdings müssen dafür durch die Erstellung einer eigenen Klasse von Daten, die nicht in das Gebiet der bekannten Werte fallen [RPP97], bei hochdimensionalen Eingaberäumen sehr viele generierte Daten gelernt werden.

Eine vergleichende Herleitung von lokalen Fehlerabschätzungen („error bars“) für lineare und nicht-lineare Regressionsmodelle mit auf Fehlerquadratsummen basierenden Kostenfunktionen von der einfachen und der multivariaten linearen Regression über verallgemeinerte lineare Modelle²⁵⁾ bis zu nicht-linearen KNNs mit mehrdimensionalen Eingaberäumen bieten [PR97, PR98]. Bei der linearen Regression lassen sich unter Verteilungsannahmen für die bedingten Mittelwerte und die Regressionskoeffizienten Konfidenzintervalle und für die Werte der Ausgabegröße Prognoseintervalle angeben [Loh01, Bor93, PTVF95, S. 689 ff.]. Bei nicht-linearen Modellen können die Fehlerbalken durch Linearisierung (*Taylor-Entwicklung*) geschätzt werden [PR98]. Für MLPs mit sigmoiden Aktivierungsfunktionen ist die Konfidenzintervall-Schätzung für die Netzwerkausgaben mit Hilfe einer linearen Approximation der Fehler (*Jacobi-Matrix*) und einer t -Verteilung möglich [CLR96].

Die Unsicherheit von Netzwerkprognosen bezüglich der Ausgabegröße setzt sich aus der Streuung in den Daten und den Limitationen des Modells zusammen. Sind beide unabhängig voneinander, lassen sich beide Streuungskomponenten zu einer Gesamtstreuung addieren [PEM00]. Nach [PR97] ist der Prognosefehler noch weiter zu zerlegen. Ein großer

²⁵⁾ Zu den „Generalized-Linear-Networks“ zählen auch gewöhnliche RBFNs, wenn die RBF-Zentren festgelegt sind.

systematischer Fehler des Modells existiert in Bereichen des Eingabedatenraumes, in denen die Netzausgaben von den zu beschreibenden bedingten Erwartungswerten stark abweichen, vgl. Abb. 2.7. Diese auch als *Modell-Bias* bezeichnete Fehlerkomponente wird genauso wie diejenige, die aus der Ungewissheit bezüglich der Eingabewerte („input noise“) resultiert, bei [PR97] als vernachlässigbar vorausgesetzt. Die nicht durch das Modell mit den verwendeten Prädiktoren erklärbare Streuung der Ausgabegröße („target noise“) in Form der Quadratfehler der Regressionsmodellprognosen wird durch ein zweites Netzwerk gelernt, vgl. den Predictive-Error-Bars-Ansatz in Unterabschnitt 2.2.1 (S. 20).

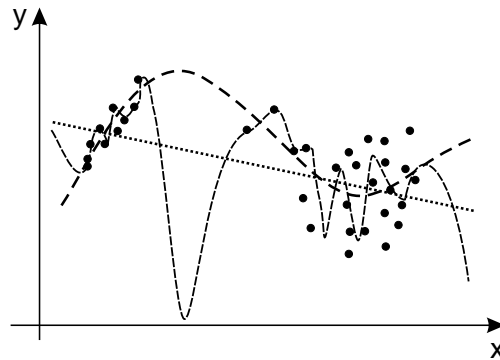


Abbildung 2.7: *Problem der Unter- und Überanpassung. Sowohl eine affine Funktion (gepunktetes Geradenstück) als auch eine zu sehr an die Trainingsdaten angepasste Funktion (dünne, gestrichelte Kurve) sind schlechte Approximationen der Regressionsfunktion (dicke, gestrichelte Kurve) und verursachen große Fehler für Testdaten.*

Vor allem in Bereichen des Eingaberaumes mit geringer Datendichte werden in Abhängigkeit von den verwendeten Trainingsdaten unterschiedliche Prognosen entstehen. Man spricht dann von der „Daten-bedingten Modell-Varianz“. Dieser Beitrag zum Fehler kann durch die sog. *Delta-Methode* näherungsweise bestimmt werden. Je nach den gewählten Startwerten der Gewichte können bei iterativen Trainingsverfahren für nicht-lineare KNNs unterschiedliche *lokale Minima* der Fehlerfunktion gefunden werden, wodurch sich eine als „Trainings-bedingte Modell-Varianz“ bezeichnete Streuung von Prognosen ergibt. Diese kann durch ein *Komitee*^(e) von Netzwerken beschrieben werden, das in [PR97] durch RBFNs mit nicht-lokalen *Thin-Plate-Spline-Funktionen*, siehe Gl. 2.4 (S. 17), gebildet wurde.²⁶⁾

Bei der Anwendung des *Bootstrap-Verfahrens* [ET93] werden aus den Trainingsdaten B Stichproben durch „Ziehen mit Zurücklegen“^(e) gewonnen, wobei die Prognose des *Bootstrap-Komitees* von KNNs aus dem arithmetischen Mittel der Ausgaben der verschiedenen Mitglieder, die Modell-Unsicherheit aus ihrer (Stichproben-)Varianz und die lokale Streuung in den Daten durch ein weiteres KNN, das die Residuen der Komitee-Prognosen lernt, berechnet werden, siehe [PEM00].

²⁶⁾ Ein Komitee von Netzwerken ist in der Regel bezüglich des Generalisierungsfehlers besser als die mittlere Modellqualität einer Menge von einzelnen Netzwerken [RPP97, Bis95].

Beispiele aus [Hes97] zeigen, dass *Ensembles* von KNNs mit Bootstrap-Replikaten der Original-Daten mit auf Validierungsdaten ermittelten Fehlerabschätzungen vor allem in Bereichen mit wenigen Daten besser geeignet sind als solche, die auf der Berechnung der Hesse-Matrix^(e) beruhen. Es wird aber vorausgesetzt, dass die KNN-Regressionsschätzung keinen systematischen Modell-Fehler enthält. Der Ansatz ist für große Datensätze sehr aufwendig [PR97]. Ensembles von verschiedenen Netzwerken wurden auch von [CCB99] zusammen mit Bootstrap-Verfahren angewendet, um Konfidenz- und Prognoseintervalle abzuschätzen. Die Idee, aus mehreren Netzen sowohl eine gemeinsame Netzwerkausgabe als auch die Streuungen der einzelnen Mitglieder zu berechnen, ist in Abb. 2.8 illustriert.

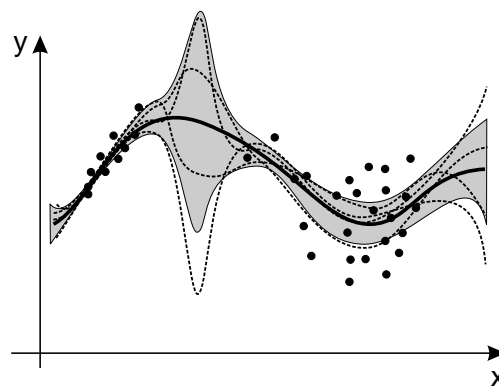


Abbildung 2.8: Veranschaulichung der auf einem Komitee oder Ensemble von Modellen basierenden Prognose und der Abschätzung ihrer Streuungen um die als Mittelwerte der Mitgliederprognosen gegebene Ausgabe.

Die lokale Zuverlässigkeit lässt sich in die beim Trainieren des KNNs zu minimierende Kostenfunktion integrieren. Eine Möglichkeit besteht darin, die Struktur vorwärts gerichteter Schichtennetze durch die Einfügung von sog. $\hat{\sigma}^2$ -Einheiten in die Ausgabeschicht und die „Update-Formeln“ für das Lernen durch Fehlerrückführung zu modifizieren. Dabei werden die Gewichte zur Anpassung des Modells an die Regression auch in Abhängigkeit von der erwarteten Streuung der Vorhersagen verändert, sodass sich das Netz beim Lernen auf die Bereiche des Eingaberaumes „konzentriert“, in denen niedrige Varianz bzw. Fehler zu erwarten sind [WN94, NW95]. Bei der Schätzung des Erwartungswertes und der Varianz der Wahrscheinlichkeitsverteilung der Zielvariablen nach [WN94] wird ein ausreichend großer Datensatz vorausgesetzt, sodass keine Überanpassung stattfindet.

Ein praktischer Methodenvergleich zur Abschätzung von Konfidenzintervallen sowohl für als „synthetisch“ bezeichnete als auch für gemessene Daten kommt zu dem Ergebnis, dass der Maximum-Likelihood-Ansatz schnell und bezüglich des verwendeten Gütemaßes ausreichend geeignet ist, während Bayes-Netze mit gaußschen Approximationen und KNNs nach dem Bootstrap-Ansatz eine lange Trainingszeit benötigen [PEM00].

Eine Gegenüberstellung der Resultate bei der Anwendung verschiedener Ansätze — u. a. die in [CLR96, LKU92a, SMZM97] — zur Schätzung der Konfidenzgrenzen von KNN-Ausgaben anhand willkürlich gewählter, einfacher „Testfunktionen“ von eindimensionalen unabhängigen Variablen findet sich in [YKC⁺00].

2.3 Hybridisierung von SOMs und RBFNs

2.3.1 Hybride Neuronale Netzwerke

Um die Vorzüge und Möglichkeiten unterschiedlicher KNNs sowie statistischer Verfahren zu nutzen, lassen sich diese durch eine sogenannte *Hybridisierung* miteinander kombinieren. Ein hybrides System kann dabei durchaus Eigenschaften aufweisen, die keiner der Bausteine allein besitzt.

Es gibt zahlreiche Beispiele für hybride Netze bzw. hybride Trainingsverfahren, darunter das klassische *Perzeptron* von ROSENBLATT [Ros62, Roj93, S. 375], die Erweiterung gewöhnlicher RBFNs mit linearen Neuronen in Gl. 2.5 (S. 18), das Training von RBF-Neuronen durch ein Clusteranalyse-Verfahren, die Vorschaltung einer *Hauptkomponentenanalyse*^(e) (PCA) vor Schichtenetzwerke zur Dimensionsreduktion des Merkmalsraumes [ABGT00], *Counterpropagation-Netze*, vgl. [Roj93, S. 367 ff.] und [Zel94, S. 189 ff.], *motorische Karten* [RMS90, S. 115] sowie stückweise lineare Funktionen [Roj93, S. 369 f.], die als *lineare Assoziatoren* und *lokale lineare Abbildungen* [Ves97, RMS90, S. 291 ff.] bekannt sind. Ein *Perzeptron-Radiale-Basisfunktionen-Netz* (PRBFN) genanntes hybrides Netzwerk besitzt in einer verdeckten Schicht nicht nur RBF-Neuronen mit einem Bias-Element, sondern zudem Perzeptron-Zellen [CI00]. Schnell lernende RBFNs lassen sich zur Prozess-Steuerung in einem hybriden System einsetzen, um den Gültigkeitsbereich miteinander konkurrierender Modelle, z. B. linearer Approximationen, abzuschätzen und über ihre Aktivität ein Maß für eine möglicherweise unzulässige Extrapolation zur Verfügung zu stellen [TKLP99].

2.3.2 RBFSOM — Anwendung einer SOM zur Positionierung von RBF-Zentren und zur Visualisierung

Neben den auf S. 18 genannten Möglichkeiten zum Positionieren der Zentren eines RBFNs besteht ein weiterer Ansatz der Anwendung von unüberwachtem Lernen in der Verwendung des SOM-Algorithmus, vgl. S. 12 (Abschnitt 2.1.2). Diese Methode findet in [Roj93, S. 370 ff.], [Bis95, S. 187 f.] und [Zel94, S. 239 f.] Erwähnung und besitzt Ähnlichkeiten zu der Darstellung in [Fri94].

Im ersten Lernschritt des im Folgenden als *RBFSOM* bezeichneten Netzes wird zunächst eine SOM mit den Eingabevektoren \mathbf{x}_i ($i = 1, \dots, n$) trainiert. Dadurch verteilen sich die Kodebuchvektoren \mathbf{z}_k ($k = 1, \dots, m$) unüberwacht im Eingabedatenraum. Identifiziert man die SOM-Kodebuchvektoren mit den RBF-Zentren, können im zweiten Lernschritt die Gewichte w_k des Ausgabeneurons mit den in Unterabschnitt 2.1.3 (S. 18) dargestellten Verfahren überwacht optimiert werden.

Über die Abb. 1.4 (S. 5) und Abb. 1.5 (S. 6) mit reellen Netzeingaben hinaus dient Abb. 2.9 im Falle eines zweidimensionalen Eingabedatenraumes der Veranschaulichung der Identifikation der SOM-Kodebuchvektoren und der RBF-Zentren sowie der Superposition von RBFs in einem RBFSOM-Netz.

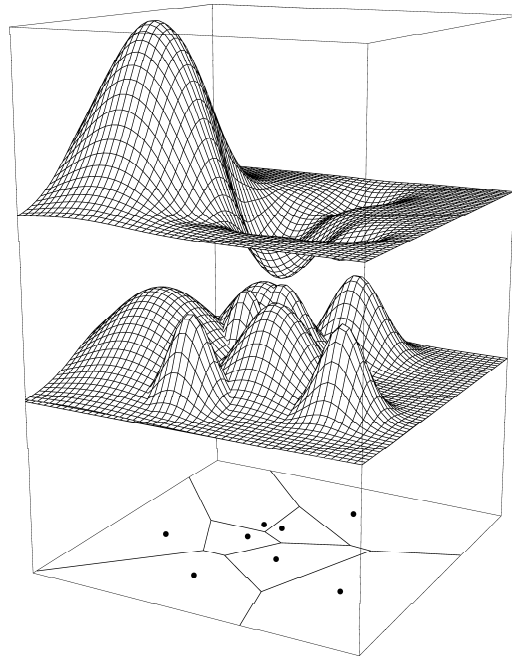


Abbildung 2.9: Voronoi-Parkettierung eines zweidimensionalen Eingabedatenraumes und Überlagerung von dreidimensionalen „Gauß-Glocken“. Auf der Grundfläche liegen acht, als Punkte dargestellte RBF-Zentren, die mit den SOM-Kodebuchvektoren identifiziert werden, und die durch sie gegebene Voronoi-Parkettierung der Ebene. Darüber befinden sich, in dieser Darstellung „angehoben“, die RBFs mit unterschiedlichen Bandbreiten positioniert. Ganz oben ist die Superposition der RBFs als Netzausgabe mit teilweise negativen Gewichten des linearen Ausgabeneurons dargestellt.

An dieser Stelle lassen sich die in Abschnitt 1.1 für eindimensionale unabhängige Größen motivierten Aussagen für mehrdimensionale Zufallsvektoren, deren Realisierungen die Netzeingaben $\mathbf{x} \in \mathbb{R}^d$ sind, formulieren. Für jede der Klassen \mathcal{X}_k^L , die sich durch die Einzugsgebiete der m SOM-Zellen ergeben, sind die Anzahlen der in ihnen liegenden Lern- und Test-Eingabedaten, Streuungsmaße wie die Varianz, die Spannweite und andere Quantile sowie Lageparameter wie das arithmetische Mittel und der Median für die zugehörigen Messwerte y_i der abhängigen Größe Y in $\mathcal{Y}_k^L := \{y_i \mid x_i \in \mathcal{X}_k^L\}$ einfach berechenbar.

Um vor einer neuartigen Netzeingabe \mathbf{x}_0 im Sinne einer Extrapolation zu warnen, kann nicht nur der maximale Abstand von \mathbf{x}_0 zu allen *Prototypen* \mathbf{z}_k , sondern auch die maximale Aktivität über alle RBF-Neuronen [LKU92a, S. 822] dienen. Die Wahl einer geeigneten

Schwelle ist in beiden Fällen problemabhängig.

Durch die gleichzeitige Verwendung des Parzen-Dichteschätzers (S. 22) und des Fehlerbalkenansatzes (S. 20) sowie eine zusätzliche Hybridisierung des RBF-SOM-Netzes mit einem VI-Netz (S. 20) sind weitere Abschätzungen der Datendichte, des lokal zu erwartenden Fehlers sowie lokaler Prognose- und Konfidenzintervalle möglich.

Eine der großen Stärken der RBF-SOM-Netzwerke stellen die zahlreichen Visualisierungsmöglichkeiten der Daten und der Aktivitäten der verdeckten RBF-Neuronen dar.

Die Software *SOM_PAK* liefert, wie in Unterabschnitt 2.1.2 ab S. 15 geschildert, Veranschaulichungen der Komponentenwerte der Kodebuchvektoren, der Trajektorien der Siegerneuronen für Sequenzen von Eingabemustern auf dem SOM-Gitter sowie der Abstände der Kodebuchvektoren durch U-Matrix- und Sammon-Abbildungen.

Eine weitere Möglichkeit besteht in der Darstellung der Muster der einzelnen SOM-Prototypen \mathbf{z}_k als Matrix von einzelnen Linien-Grafiken der Werte der Vektorkomponenten über ihren Indizes in der Anordnung des SOM-Gitters (Abb. 4.11, S. 86). Alternativ können in den Einzelgrafiken auch Balkendiagramme erzeugt werden, welche die Muster für den Betrachter oft deutlicher erkennbar werden lassen (Abb. 4.12, S. 87).

Zahlreiche Visualisierungen verschiedener Aspekte der Daten und der Netzwerkaktivitäten lassen sich mit Hilfe des SOM-Gitters als verdeckter Schicht des RBF-SOM-Netzes durch Graustufen anzeigen. Im Rahmen dieser Arbeit gilt — von wenigen, ausdrücklich angegebenen Ausnahmen abgesehen —, dass die darzustellenden Werte umso kleiner sind, je dunkler die betreffende Graustufe ist und insbesondere die niedrigsten Werte durch Schwarz und die Maximalwerte durch Weiß repräsentiert werden. Ganzzahlige Werte lassen sich auf den Zellen zusätzlich explizit angeben. Beispiele für diese Möglichkeiten sind: die Anzahlen $\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$ und $\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^T$ der in den Einzugsgebieten der einzelnen SOM-Neuronen liegenden Lern- (Abb. 4.15, S. 88) bzw. Testdaten (Abb. 4.31, S. 96), die durch die Testdaten geschätzten, in den SOM-Einzugsgebieten durchschnittlich zu erwartenden Abweichungen zwischen Prognosen und Messwerten (Abb. 4.32, S. 97), die den Grad der Bekanntheit für beliebige Eingabevektoren veranschaulichenden euklidischen Abstände zu den Kodebuchvektoren der SOM (Abb. 4.33, S. 97) sowie die ausnahmsweise durch invertierte Graustufen wiedergegebenen Aktivitäten der einzelnen RBF-Neuronen (Abb. 4.34, S. 98), die Veranschaulichungen der Likelihoods des Parzen-Dichteschätzers und des VI-Netzes für die Kodebuchvektoren als Hinweise auf die Datendichte im Eingabedatenraum (Abb. 4.16, S. 88), Spannweiten $\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L) - \min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ als Streuungsmaß der Werte y_i , deren zugehörige Eingabemuster \mathbf{x}_i in die Einzugsgebiete der entsprechenden SOM-Neuronen fallen (Abb. 4.17, S. 89) und schließlich nach einer Kreuzvalidierung die zu erwartenden Prognose- und Konfidenzintervalle durch die VI-Netz-Ausgaben für die Repräsentanten der einzelnen SOM-Klassen (Abb. 4.18, S. 90).

Für die Veranschaulichung der Verteilung der Residuen $y(t) - \hat{y}(t)$ dient ein *Histogramm* als Balkendiagramm (Abb. 4.20, S. 91).

Hält man für die Prognosen der Testdaten jeweils eine Netzeingabegröße beispielsweise auf ihrem Mittelwert konstant, sollten sich auf diese Weise für die wichtigsten Prädiktoren die größten Fehler ergeben, während irrelevante Netzeingaben wenn sie auf einem

festen Wert gehalten werden, kaum eine Verschlechterung der Modellgüte verursachen. Auf diese Weise lassen sich die Wichtigkeiten der Prädiktoren für das KNN-Modell über die Erhöhung des RMSEs abschätzen und z. B. in einer Balkengrafik veranschaulichen (Abb. 4.46, S. 105).

Durch *Streudiagramme*^(e) lassen sich die prognostizierten Werte den Messwerten (Abb. 4.19, S. 90) und für Hinweise auf Homo- bzw. Heteroskedastizität den Residuen (Abb. 4.22, S. 92) sowie die Beträge der Residuen den minimalen euklidischen Abständen der Testeingabevektoren zu den Kodebuchvektoren (Abb. 4.25, S. 93) gegenüberstellen.

In Form von Zeitreihendarstellungen als Liniengrafiken sind die folgenden Größen für jeden Testdatensatz über seiner Nummer zu veranschaulichen: die Netzwerkprognosen im Vergleich zu den Messwerten (Abb. 4.19, S. 90), die Residuen (Abb. 4.21, S. 91), die euklidischen Abständen zwischen den Testeingabevektoren und den jeweils nächstliegenden Kodebuchvektoren bzw. RBF-Zentren (Abb. 4.23, S. 92), die maximale Aktivität aller RBF-Neuronen als Maß für den Grad der Extrapolation (Abb. 4.24, S. 93), die Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$ von Lerndaten, die im Einzugsgebiet der SOM-Neuronen die Netzwerkprognosen in diesem Bereich stützen könnten (Abb. 4.26, S. 94), die Spannweiten der y_i in den einzelnen SOM-Klassen (Abb. 4.27, S. 94) und die arithmetischen Mittel $\bar{y}_{\kappa(\mathbf{x}_i)}$ der Elemente von $\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L$ als ein alternatives Regressionsmodell (Abb. 4.28, S. 95).

Zusätzlich lassen sich für die Messwerte der Test-Ausgabedaten, ggf. gemeinsam mit den Prognosen, durch Zeitreihendarstellungen mit grau hinterlegten Intervallen illustrieren, die etwa durch die Minima $\min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ und Maxima $\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ der Ausgabegröße innerhalb der SOM-Klassen bezüglich der Lerndaten gegeben sind (Abb. 4.29, S. 95), oder der Fehlerbalken (Predictive-Error-Bars), die durch die oben beschriebene Kreuzvalidierung bestimmt wurden (Abb. 4.30, S. 96).

Auch für hochdimensionale Eingaberäume kann die Approximation der Regressionsfunktion durch das RBF-SOM-Netz veranschaulicht werden, indem nur eine oder zwei Eingabegrößen in den durch die Lerndaten gegebenen Wertebereichen variiert und die restlichen konstant auf willkürlich gewählten oder etwa ihren mittleren Werten gehalten werden. Beispiele für Darstellungen dreidimensionaler Graphen der durch das KNN vermittelten Funktion für teilweise konstante Variable sind in den Abbildungen 4.3 und 4.36 (S. 71 und S. 99) zu sehen. Gemeinsam mit der partiellen Netzausgabefunktion in Abhängigkeit von nur einer veränderlichen und sonst konstant gehaltenen Größen lassen sich einfacher Maße der Zuverlässigkeit illustrieren, z. B. die maximalen und die mittleren Aktivitäten der RBF-Neuronen, vgl. Abb. 4.38 (S. 100) und Abb. 4.41 (S. 102), die Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sowie die Größe $\#\mathcal{X}_{\kappa(\mathbf{x})}^L$, vgl. Abbildungen 4.39 und 4.42 (S. 101 und 103), die schon oben genannten Spannweiten der Ausgabegröße für Eingaben innerhalb der SOM-Neuronen-Einzugsgebiete und die erwarteten Fehlerbalken nach dem Predictive-Error-Bars-Ansatz (Abbildungen 4.40 und 4.43 auf S. 102 und S. 103) und die Schätzungen der Konfidenz- und Prognoseintervalle durch das VI-Netz (Abb. 4.44, S. 104).

2.3.3 Reskalierung von Prädiktoren durch überwachtes Lernen

Auf radial basierten Netzwerken wie RBFNs und GRNNs, bei denen die Aktivitäten der verdeckten Neuronen und damit die Netzausgaben von den Abständen der Eingabedatenpunkte von den im Netz gespeicherten Prototypen abhängen, lastet der „Fluch der Dimensionen“^(e) [Bel61, Bis95, S. 7 ff. und S. 184]. Durch das exponentielle Anwachsen der abzubildenden Hypervolumen in Abhängigkeit von der Dimension des Eingaberaumes [Sar02] sind mit jedem hinzukommenden Prädiktor weitere Ressourcen, z. B. in Form von Neuronen und ihren Gewichten, und damit mehr Zeit sowohl in der Trainings- als auch in der Arbeitsphase des Netzwerkes nötig. Irrelevante Eingabemerkmale sind daher aus dem Modell zu entfernen [Oba98] und für weitgehend redundante Größen bietet sich alternativ die Möglichkeit, sie beispielsweise durch eine *Faktoren-* bzw. *Hauptkomponentenanalyse* [BEPW96, S. 189 ff. und S. 223] oder *Flaschenhalsnetze* [Dap98, S. 118 ff.] zusammenzufassen. Allerdings sind selten vollkommen redundante oder irrelevante Merkmale zu erwarten, sondern in der Regel geht eine Dimensionsreduktion auch mit einem Informationsverlust einher.

Weiterhin besteht bei radialen Basisfunktionen das Problem, dass durch die willkürliche Wahl der (Re-)Skalierung der Eingabegrößen in der Vorverarbeitungsphase die Güte der Ausgabe des Netzes beeinflusst wird und eine günstiger Skalierungsfaktor gefunden werden muss. Der im Folgenden vorgestellte Ansatz versucht, die Reskalierung der Netzeingaben zu optimieren.

Dafür wird eine neue Eingabeschicht von d Neuronen vorgeschaltet, wobei für jedes von ihnen seine Ausgabe x_j allein an das ursprüngliche Eingabeneuron mit dem Index j weiterleitet, das nun ein Gewicht s_j besitzt. Mit diesen Skalierungsfaktoren s_j $j = 1, \dots, d$ lautet die Eingabe in die radiale Schicht des modifizierten RBF-Netzes $\mathbf{x}^s := (s_1 x_1, s_2 x_2, \dots, s_d x_d)^{27)}$ und aus Gl. 2.2 (S. 17) wird

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^m w_k \phi_k(\mathbf{x}^s) + w_0. \quad (2.13)$$

Die Idee gleicht dem in [Dap98] allgemein vorgestellten und speziell auf MLPs angewendeten Ansatz, bei dessen Darstellung die den Skalierungsfaktoren s_j entsprechenden Größen als *Sensitivitäten* bezeichnet werden. Die Verwendung einer vorgeschalteten, linearen Schicht für RBFNs ist als Transformation der radialen Basisfunktionen in *elliptische Basisfunktionen* interpretierbar, vgl. [Sar02].

Grundsätzlich kann die Optimierung der Bandbreiten und der RBF-Zentren durch die Anwendung der *Fehlerrückführung*^(e) und beispielsweise einem *Gradientenabstiegsverfahren* vollständig überwacht erfolgen. Entsprechende Hinweise sind für RBFNs mit variablen Bandbreiten in [Bis95, S. 190] und für verallgemeinerte *Hyper-Basisfunktionen-Netze* in [Zel94, S. 235 ff.] angegeben. Die Bandbreiten werden hier — im Unterschied zu [Bis95, S. 190 f.] — nicht einzeln verändert, sondern bleiben konstant, da sie durch das Training sehr groß werden können [Bis95, S. 191].

²⁷⁾ Diese Reskalierung lässt sich auch als Anwendung der Diagonalmatrix mit s_j als Elemente in der Diagonalen und sonst Nullen auffassen.

Für RBFNs mit gaußschen RBFs wurden folgende Modifikationen zur automatischen Re-skalierung der Prädiktoren vorgenommen. Als zu minimierende Kostenfunktion in Abhängigkeit der Neuronengewichte s_j , z_{kj} und w_k lässt sich die halbe Fehlerquadratsumme $E = \frac{1}{2} E_{\text{SQE}}$, vgl. Gl. 1.2 (S. 3), wählen. Der Wert von E ergibt sich als Summe der Einzelfehler $E_i := \frac{1}{2} (y_i - \hat{y}_i)^2$ über alle Trainingsmuster (\mathbf{x}_i, y_i) ($i = 1, \dots, n$). Mit dem während des Lernens rückwärts durch das Netzwerk geleiteten Fehlerstrom

$$\begin{aligned}\delta_i^w &:= -(y_i - \hat{y}_i) \\ \delta_i^{z_k} &:= \delta_i^w w_k \phi_k(\mathbf{x}_i^s) \\ \delta_i^{s_j} &:= -\sum_{k=1}^m \delta_i^{z_k} \frac{s_j x_{ij} - z_{kj}}{h_k^2}\end{aligned}\tag{2.14}$$

haben die partiellen Ableitungen die Form

$$\begin{aligned}\frac{\partial E_i}{\partial w_k} &= \delta_i^w \phi_k(\mathbf{x}_i^s) \\ \frac{\partial E_i}{\partial z_{kj}} &= \delta_i^{z_k} \frac{s_j x_{ij} - z_{kj}}{h_k^2} \\ \frac{\partial E_i}{\partial s_j} &= x_{ij} \delta_i^{s_j}.\end{aligned}\tag{2.15}$$

Der Gradientenabstieg durch ein *Gesamtschrittverfahren*,²⁸⁾ bei dem der Fehler E vor jedem Lernschritt t für alle Trainingsmuster (\mathbf{x}_i, y_i) berechnet wird, lässt sich durch die Verwendung eines *Momentum-Terms*, vgl. [Bis95, S. 267] und [Zel94, S. 239], beschleunigen:

$$\begin{aligned}\Delta w_k(t) &= -\eta_w \sum_{i=1}^n \frac{\partial E_i}{\partial w_k} + \mu \Delta w_k(t-1) \\ \Delta z_{kj}(t) &= -\eta_z \sum_{i=1}^n \frac{\partial E_i}{\partial z_{kj}} + \mu \Delta z_{kj}(t-1) \\ \Delta s_j(t) &= -\eta_s \sum_{i=1}^n \frac{\partial E_i}{\partial s_j} + \mu \Delta s_j(t-1).\end{aligned}\tag{2.16}$$

Wegen der Heterogenität der Neuronen in den einzelnen Schichten können unterschiedliche Lernraten η_s , η_z und η_w gewählt werden. Der mit der Bedingung $0 \leq \mu \leq 1$ möglichst günstig zu wählende Faktor μ ist im Rahmen dieser Arbeit für alle Gewichte gleich.²⁹⁾ Setze zur Initialisierung $\Delta w_k(0) = \Delta z_{kj}(0) = \Delta s_j(0) = 0$. Die Netzwerkparameter werden bei jedem Lernschritt in Anlehnung an [Bis95, S. 255] folgendermaßen verändert:

$$\begin{aligned}w_k(t+1) &= w_k(t) + \Delta w_k(t) \\ z_{kj}(t+1) &= z_{kj}(t) + \Delta z_{kj}(t) \\ s_j(t+1) &= s_j(t) + \Delta s_j(t).\end{aligned}\tag{2.17}$$

²⁸⁾ Weitere Bezeichnungen sind *Batch-* und *Offline-Verfahren* [Zel94, S. 107].

²⁹⁾ Durch den Momentum-Term wird eine Trägheit bei der Bewegung durch den Gewichtsraum hinzugefügt [Bis95, S. 267]. Die Werte für μ liegen „am häufigsten“ zwischen 0.6 und 0.9 [Zel94, S. 115].

Auf die Anwendung von Methoden zur Steigerung der Geschwindigkeit bei der Optimierung der Gewichte wie eine *Schrittweitensteuerung* [Dap98, S. 11] oder das *Levenberg-Marquardt-Verfahren* [Bis95, S. 290 ff.] wurde zunächst verzichtet.

Um den iterativen Gradientenabstieg mit der Fehlerrückführung hauptsächlich zum „Feintuning“ zu verwenden, sind $s_j(0) = 1$ für $j = 1, \dots, d$, durch die beschriebenen unüberwachten Verfahren festgelegte RBF-Zentren $\mathbf{z}_k(0)$ und durch schnelle numerische Methoden (vgl. S. 18) gefundene $w_k(0)$ nach einer auf S. 36 beschriebenen $[0,1]$ -Skalierung oder z-Transformation der unabhängigen Größen geeignete Startwerte. Das Verfahren kann nach einer vorgegebenen Anzahl von Lernschritten, beim Unterschreiten des Lernfehlers E unter eine bestimmte Schranke oder falls ein ansteigender Fehler auf Validierungsdaten bei weiterhin fallendem Lernfehler einen Hinweis auf Überanpassung liefert beendet werden.

Die Vorteile eines RBFNs mit einer solchen Reskalierungsschicht gegenüber einem gewöhnlichen RBFN verdeutlicht Abb. 2.10. In dem Anwendungsbeispiel wurde die sinu-

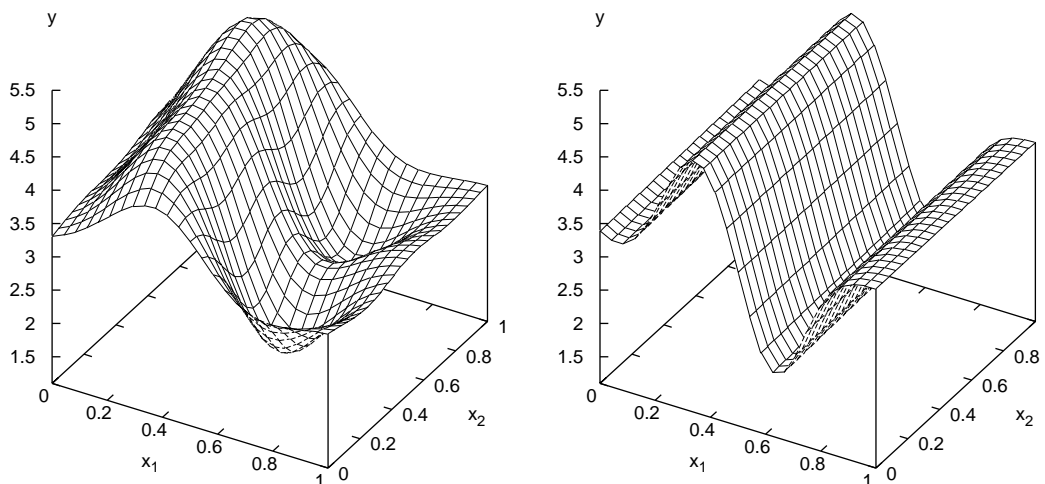


Abbildung 2.10: Approximation einer sinusoiden Funktion, links durch ein gewöhnliches RBFN mit neun RBF-Neuronen, rechts durch ein RBFN mit einer durch Fehlerrückführung und Gradientenabstieg trainierten Reskalierungsschicht und nur drei RBF-Neuronen. Zu den Funktionswerten $\tilde{y}(x_1, x_2) = 1.5 \sin(3\pi(x_1 + 0.5)) + 3.5$ der zu nähernden Funktion \tilde{y} wurde ein schwaches, normalverteiltes Rauschen addiert; die zufällig gewählten Netzeingaben $(x_1, x_2) \in [0, 1]^2$ lassen sich für x_1 als Winkel und für x_2 als irrelevante Eingabe (Rauschen) interpretieren. Die 70 Lern-Eingabedaten stammten aus dem Intervall $[0.15, 0.80] \times [0.01, 0.98]$.

soide Funktion $\tilde{y}: \mathbb{R}^2 \rightarrow \mathbb{R}$, mit $\tilde{y}(x_1, x_2) = 1.5 \sin(3\pi(x_1 + 0.5)) + 3.5$, zu deren Funktionswerten ein schwaches, normalverteiltes Rauschen addiert worden war, durch ein gewöhnliches RBFN mit neun RBF-Neuronen und ein RBFN mit einer durch Fehler-rückführung und Gradientenabstieg trainierten Reskalierungsschicht und nur drei RBF-Neuronen approximiert. Beide Netze wurden mit der in Kapitel 3 dokumentierten Soft-

ware *Visualrbfn* simuliert. Die 70 Lern-Eingabedaten stammten aufgrund der zufälligen Auswahl der Trainingspunkte aus dem Intervall $[0.15, 0.80] \times [0.01, 0.98]$. Die Bandbreiten wurden auf 0.2 festgesetzt und die Initialisierungswerte der Gewichte $s_j(0)$ zur Reskalierung waren für $j = 1, \dots, d$ gleich eins. Nach dem Training beider RBFNs mit Hilfe einer SOM und einer Singulärwertzerlegung wurde das Netz mit weniger verdeckten Neuronen und der Reskalierungsschicht mit 150 Lernschritten nach dem oben beschriebenen Verfahren (ohne Momentum-Term) nachtrainiert. Die Skalierungsgewichte betragen anschließend $s_1 = 1.9047$ und $s_2 = -0.0076$. In diesem Fall waren deutlich weniger RBFs nötig, um die Funktion \tilde{y} mit einer besseren Generalisierungsqualität zu approximieren. Nach dem hier beschriebenen überwachten Training lassen sich solche Eingabegrößen, deren Skalierungsfaktor betragsmäßig unterhalb einer vorzugebenden Schranke liegt und die daher ein geringes Gewicht aufweisen, aus dem Modell entfernen. Dieses Vorgehen kann als Spezialfall von sog. *Pruning* [Bis95, S. 354] bezeichnet werden.

Das — selbst bei der Anwendung iterativer Verfahren wie der Positionierung der RBF-Zentren mit Hilfe einer SOM — im Vergleich zu MLPs schnelle Lernen der RBFNs kann durch Gradientenabstiegsverfahren sehr viel Zeit in Anspruch nehmen. Das größte prinzipielle Problem des hier geschilderten Ansatzes ist durch *lokale Minima* in der Fehlerlandschaft gegeben, in denen Gradientenabstiegsverfahren „stecken“ bleiben können und dann ein³⁰⁾ gesuchtes *globales Minimum* der Fehlerfunktion E nicht gefunden wird. Gegebenenfalls muss das Training mit unterschiedlichen Startwerten mehrfach wiederholt werden.

Für die Anwendung des Gradientenabstiegsverfahrens müssen weder die Werte der Eingabegrößen reskaliert noch die Positionen der Zentren der RBFs verändert werden. Will man beispielsweise nur die Ausgabeschicht trainieren, weil eine Singulärwertzerlegung versagt, setze man die Lernraten η_s und η_z gleich null.

Um die in Unterabschnitt 2.3.2 (S. 26) genannten Zuverlässigkeitsmaße und Visualisierungsmöglichkeiten nutzen zu können, lassen sich günstige Faktoren zur Reskalierung mit einer Teilmenge der Lerndaten ermitteln, anschließend alle — oder auch nur die verbleibenden — Lern- und Testdaten entsprechend transformieren und dann ein RBFSOM-Netzwerk anwenden. Gleichwohl lässt sich eine automatische Reskalierung von Prädiktoren und allgemeiner das überwachte Lernen von RBFNs nicht immer mit der Anwendung von RBFSOM-Netzen in Einklang bringen, wie Abb. 2.11 verdeutlicht. Die Approximation der affinen Funktion $y(x) = 0.5x + 0.1$ auf der Basis von 10 Punkten im Intervall $[0.45, 1.35]$ durch ein RBFSOM mit 3 gaußschen RBFs (Bandbreiten $h = 0.2$) und einem Bias-Element sowie einem RBFN mit je einem Reskalierungs-, einem RBF- (gaußsche RBF) und einem Ausgabeneuron, aber ohne Bias-Element zeigt,³¹⁾ dass ein unüberwachtes Lernen für die Neuronen der verdeckten Schicht eine im Vergleich zu vollständig überwachten Verfahren schlechtere Generalisierungsfähigkeit und eine höhere Netzwerk-

³⁰⁾ Man kann nicht grundsätzlich davon ausgehen, dass es nur ein einziges globales Minimum der Fehlerfunktion gibt, daher wurde hier der unbestimmte Artikel verwendet.

³¹⁾ Das überwachte Lernen des RBFNs wurde nach 400 Lernschritten mit η_s, η_z und η_w gleich 0.003 sowie $\mu = 0.7$ beendet.

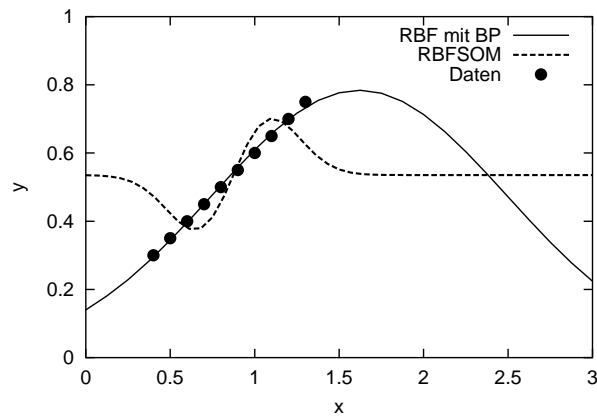


Abbildung 2.11: Vergleich der Approximation einer affinen Funktion durch ein RBFSOM-Netz, das über 3 verdeckte Neuronen und ein Bias-Element verfügt, und ein RBFN, das kein Bias-Element und nur eine einzige RBF-Zelle besitzt, wobei die Ausdehnung und die Position der RBF sowie das Gewicht des Ausgabeneurons durch überwachtes Nachtraining mit einer Fehlerrückführung (BP steht für Back-propagation) und einem Gradientenabstiegsverfahren bestimmt sind.

komplexität nach sich ziehen kann. Allerdings ist auch zu sehen, dass RBF-Zentren bei dem vollständig überwachten Training die Punktwolke der Eingabedaten verlassen können. In diesem Fall gibt die Aktivität der RBF-Neuronen keinen Aufschluss mehr über die Nähe zu bekannten Punkten. Nach dem überwachten Lernen zur Gewinnung reskalierter Prädiktoren ist daher ein Neutrainning eines RBFSOMs erforderlich, wenn dessen Vorzüge genutzt werden sollen. Ansonsten ist ein weiterer Ansatz vorstellbar, der grundsätzlich eine Ähnlichkeit zu der Verwendung einer SOM zur Einschätzung der lokalen Prognosegüte global approximierender MLPs [CD98] aufweist. In diesem Fall könnte eine Spezialisierung der Aufgaben darin bestehen, dass ein RBFN oder ein anderes KNN für die optimierte und effiziente Approximation der Regressionsfunktion zuständig ist und ein RBFSOM-Netz als Extrapolationsdetektor und Dichteschätzer und zur Prognose von erwarteten Fehlern und Streuungen dient.

3 Simulationssoftware

Das RBFSOM-Netzwerkmodell lässt sich mit dem vom Verfasser im Rahmen dieser Arbeit erstellten Softwarepaket *Visualrbfn* auf Computern simulieren. Es liegt mit zahlreichen Erweiterungen als eine Sammlung von Programmen in verschiedenen Programmiersprachen sowie Shell-Skripten vor.

Der folgende Abschnitt erläutert die Anwendung der einzelnen Komponenten von der Datenvorverarbeitung über die Beschreibung des Datenformates, die verschiedenen Betriebsmodi, die Steuerung des Programmablaufes durch Parameterdateien bis zu durch den Benutzer modifizierbare Shell- und *Octave*-Skripten. Weiterhin werden die Ausgaben des Programms auf der Standardausgabe und in Form von Text- und Grafikdateien dokumentiert sowie Abschätzungen der Laufzeit der Software und weitere Hinweise gegeben. Der sich daran anschließende Abschnitt 3.2 behandelt vorwiegend technische Details, beispielsweise zu der vorausgesetzten Software, der Installation von *Visualrbfn* und weiteren Optionen sowie einem Werkzeug zur Visualisierung der von dem RBFSOM-Netzwerk gelieferten Informationen. Abschließend werden einige Aspekte der Software diskutiert.

3.1 Bedienung und Benutzerschnittstelle

Die einzelnen Programme der Simulationssoftware *Visualrbfn* startet man über die Eingabeaufforderung einer interaktiven Shell oder mit Hilfe von Shell-Skripten,¹⁾ welche die Durchführung eines Experimentes mit *Visualrbfn* vereinfachen und als ein Beitrag zur Dokumentation der Versuche betrachtet werden können.

Es empfiehlt sich, für jedes Simulationsexperiment ein eigenes Verzeichnis anzulegen. Bequemerweise können die Shell-Skripten und die Optionen für das Modell enthaltende Parameterdateien vorangegangener Experimente in das Arbeitsverzeichnis kopiert und modifiziert werden.

3.1.1 Datenvorverarbeitung

In vielen Fällen sind *Rohdaten*^(e) als Eingaben für KNNs ungeeignet und zunächst einer mitunter aufwendigen *Vorverarbeitung* zu unterziehen.²⁾ Diese besteht in der Regel aus

¹⁾ In der Regel benutzt man die „Bourne Shell“ [Gt92, S. 3-3 ff.] oder andere unter *UNIX*, *Linux* und *Cygwin* bekannte Shells. Im Folgenden sind mit Shell-Skripten zwar auch Batch-Dateien für *MS-DOS* und die *MS-Windows*-Befehlsinterpreter gemeint, allerdings sind damit die Möglichkeiten der Software eingeschränkt.

²⁾ In [Oba98, S. 25 ff.] ist eine anschauliche Darstellung wichtiger Vorverarbeitungsschritte zu finden.

einer Konvertierung des Datenformates, dem Ersetzen von fehlenden oder fehlerhaften Messwerten (z. B. durch Interpolation oder Einsetzen des Mittelwertes), einer Auswahl von Objekten (etwa zur Aufteilung der Daten zum Trainieren und zum Testen der Generalisierungsfähigkeit) und dem Kodieren von dichotomen oder anderen nominal skalierten Merkmalen. Vor allem für die in dieser Dissertation verwendeten Netzwerktypen, bei denen der (euklidische) Abstand zwischen Datenpunkten eine wesentliche Rolle spielt, ist neben einer Eliminierung von *Ausreißern* (fehlerhafte, oftmals extreme Werte), einer abschwächenden Transformation von *Außenpunkten* (sehr ungewöhnliche, extreme Messergebnisse), einer Dimensionsreduktion durch eine Selektion oder eine Zusammenfassung relevanter Prädiktoren (vgl. die Ausführungen zum „Fluch der Dimensionen“ auf S. 30 sowie [Dap98]) vor allen Dingen eine *Reskalierung* der unabhängigen Größen anzuraten. Eine Möglichkeit besteht darin, für die Werte der einzelnen Variablen eine *Standardisierung* in Form einer *z-Transformation* [Bor93, S. 45] durchzuführen, bei der von jedem Wert eines Merkmals der zugehörige Mittelwert subtrahiert und diese Differenz durch die Standardabweichung dividiert wird, sodass der Mittelwert der transformierten Größe verschwindet und ihre Varianz gleich eins ist. Damit lassen sich die Abweichungen vom Mittelwert für verschiedene Merkmale vergleichen. Eine affine *Skalentransformation* (mit positiver Steigung) auf das Intervall $[0,1]$, sodass der minimale Messwert jeder Größe auf null und ihr maximaler Wert auf eins abgebildet wird und somit die *Spannweiten* einheitlich sind, ist ebenfalls geeignet. Die resultierende Größe und ihre Werte werden im Folgenden kurz als *[0,1]-skaliert* bezeichnet.³⁾ Gelegentlich ist in dieser Arbeit von einem Wert v *relativierten* Größen und Werten die Rede. Damit ist das Verhältnis der Größe bzw. des Wertes zu v gemeint. Wird v nicht erwähnt, ergibt sich ein relativierter Wert aus der Division des ursprünglichen Wertes durch das Maximum der vorliegenden Werte der jeweiligen Größe.

Die genannten Vorverarbeitungsschritte können beispielsweise mit *Excel*, *Awk*, *Perl* oder mit den in der Forschungsgruppe Neuronale Netzwerke an der Universität Kassel entwickelten *EDL*-Klienten [Klöß94] durchgeführt werden. Zur Auswahl relevanter Prädiktoren für RBFNs mit lokalen RBFs empfiehlt sich die Verwendung des vom Verfasser programmierten *EDL*-Klienten *GRNN* [Oba98], der ein gleichnamiges KNN (siehe S. 19) simuliert. Bei der *Vorwärtsmethode* werden sukzessive die Prädiktoren in das Modell aufgenommen, die eine größtmögliche Verbesserung der Modellgüte liefern, während die *Rückwärtsmethode* bei jedem Schritt diejenige unabhängige Größe entfernt, für die sich die Modellgüte am wenigsten verschlechtert. Die Kombination aus beiden wird als *schrittweise Methode* bezeichnet. Dabei kann nach der Hinzunahme einer Variablen eine andere entfernt werden [Oba98]. Zusätzlich ist die Selektion relevanter Prädiktoren auch mit *Genetischen Algorithmen* [Gol89, Oba98] möglich.

Vereinfachend werden im Folgenden auch die vorverarbeiteten Werte als *Messwerte* bezeichnet und für sie die gleichen Symbole x_i und y_i wie für die Rohdaten verwendet.

³⁾ In der Fachliteratur wird anstelle von $[0,1]$ -Skalierung gelegentlich der Begriff *Normierung* verwendet. In dieser Arbeit wird er nicht gebraucht, um Verwechslungen mit der Normierung von Vektoren auf die Länge eins zu vermeiden.

3.1.2 Format von Musterdateien

Die von *Visualrbfn* zu verarbeitenden und die ausgegebenen Daten werden in Klartext-Dateien (ASCII) gespeichert.⁴⁾ Die Werte der einzelnen Untersuchungsobjekte sind als Fließkomma-Zahlen mit Punkten als Dezimaltrenner in Zeilen angeordnet, die durch das Linefeed-Zeichen (ASCII: 10) abgeschlossen werden.⁵⁾ Die gemessenen Größen stehen in durch Tabulatoren getrennten Spalten.⁶⁾

Die erste Zeile der Datei enthält als positive Ganzzahl die Anzahl von Daten-Spalten, d. h. in der Regel die Summe der Dimensionen des Eingabe- und des Ausgabedatenraumes.

Um die Voreinstellungen der Software *Visualrbfn* verwenden zu können, wird empfohlen, die Werte der abhängigen Größe in die letzte Datenspalte zu schreiben.

Für Dateien, die Lern- oder Testdaten enthalten, kann das Zeichen # zu Beginn einer Zeile eine Kommentarzeile einleiten, die von *Visualrbfn* ignoriert wird. Damit lässt sich beispielsweise ein Tabellenkopf mit den Bezeichnern der einzelnen Merkmale einfügen. Bezeichner für die einzelnen *Muster* in den Zeilen dürfen in der letzten Spalte angegeben werden.

Das folgende Beispiel illustriert das beschriebene Format. Dabei liegen fünf Messwert-Vektoren von drei Merkmalen (X_1 , X_2 und Y) der Objekte mit den Bezeichnungen „Falle A“ bis „Falle E“ vor. Das Symbol \rightarrow steht für das Tabulator-Zeichen (ASCII: 9).

```

3
#Beispiel einer Musterdatei
#X1  →  X2      →  Y          →  Bezeichnung
5.6  →  7.90    →  12.51    →  Falle A
4.3  →  9.18    →  -1.72    →  Falle B
3.7  →  3.31    →  -4.45    →  Falle C
2.1  →  4.72    →  -6.16    →  Falle D
1.4  →  5.48    →  -5.81    →  Falle E

```

Eine Datei, die ein von *Visualrbfn* lesbares Format aufweist und die mit dem RBF-SOM-Netz zu verarbeitende Muster enthält, wird im Folgenden als *Musterdatei* bezeichnet.

Die Formate der von *Visualrbfn* und den von ihm aufgerufenen Programmen erzeugten Dateien sind in dem Unterabschnitt 3.1.6 ab S. 51 genannt.

⁴⁾ *ASCII* steht als Abkürzung für *American Standard Code for Information Interchange*. Alle in dieser Arbeit genannten ASCII-Werte sind im Dezimalsystem angegeben.

⁵⁾ Diese Markierung für das Zeilenende (EOL) in Textdateien im „UNIX-Format“ unterscheidet sich von dem im „MS-DOS-Format“ (Carriage Return und Linefeed, ASCII-Werte 13 und 10) und von dem im Format von Macintosh-Textdateien (ASCII: 13).

⁶⁾ Die für das Hauptprogramm verwendete Programmiersprache *Octave* repräsentiert alle numerischen Basisgrößen in Double-Präzision [Eat97, S. 29]. Auf den Personal-Computern des Autors mit den Betriebssystemen *Windows 98*, *Windows 2000*, *Windows XP* und *Linux* liegen die darstellbaren Fließkommazahlen zwischen $\text{realmin} = 2.2251 \cdot 10^{-308}$ und $\text{realmax} = 1.7977 \cdot 10^{308}$.

3.1.3 Programmausführung

Die Syntax für das Starten des Hauptprogramms *visualrbfn.oct* in einer Kommandozeile lautet⁷⁾

```
visualrbfn.oct [Parameterdateiname]

visualrbfn.oct [-Modus [Musterdateiname [Eingabespalten
[Ausgabespalten]]]]

visualrbfn.oct [-V [Musterdateiname [Eingabespalten
[Ausgabespalten [Gitterpunkteanzahl
[Eingabespalte1 [Eingabespalte2]]]]]]]]]
```

Das Programm *visualrbfn.oct* lässt sich in acht verschiedenen *Betriebsmodi* ausführen, die sich gemäß Tab. 3.1 über den ersten Kommandozeilenparameter *-Modus* oder in einer in Unterabschnitt 3.1.4 (S. 40) beschriebenen Parameterdatei auswählen lassen. Dabei wird bei der Angabe mehrerer Betriebsmodi nur der erste Kommandozeilenparameter oder die letzte Zuweisung in einer Parameterdatei berücksichtigt.

Zum Beispiel wird das Training eines RBF-SOM-Netzwerkes mit den in der Musterdatei *lern.dat* gespeicherten Daten und den Voreinstellungen durch die Kommandozeile

```
visualrbfn.oct -L lern.dat
```

gestartet.

Bei einem Programmstart ohne die Angabe eines Kommandozeilenparameters wird die Hilfe aufgerufen. Beginnt der erste übergebene Parameter nicht mit dem Trennstrich bzw. Minuszeichen (-, ASCII: 45), wird angenommen, dass es sich dabei um den Namen einer Parameterdatei handelt. Falls darin nichts anderes ausgewählt ist, arbeitet das Programm im Lernmodus. Mit *Musterdateiname* ist der Name der Musterdatei gemeint, welche die Lern- oder Testdaten in dem im vorigen Unterabschnitt 3.1.2 beschriebenen Dateiformat enthält. Die Bezeichner *Eingabespalten* und *Ausgabespalten* stehen für die Spaltennummern der Musterdatei, in denen Netzeingabevariable bzw. die Ausgabegröße für das RBFN stehen. In der Version 4.5 von *Visualrbfn* wird nur eine eindimensionale

⁷⁾ In eckigen Klammern [] stehende Ausdrücke sind optionale Elemente; die Klammern selbst werden aber nicht eingegeben. Zwischen einer Option und ihrem Argument darf kein Leerzeichen stehen. Um schwer verständliche Angaben mit Entweder-Oder-Alternativen der Kommandozeilenparameter zu vermeiden, ist die Darstellung der Syntax in die drei angegebenen Fälle gegliedert.

Falls man *MS-DOS* oder einen *MS-Windows*-Befehlsinterpreter benutzt und nicht über *Cygwin* verfügt, kann man die Batch-Datei *visrbfn.bat* aufrufen. Allerdings werden dabei nicht alle Möglichkeiten der Software unterstützt. Unter *MS-Windows* kann man zudem die Dateinamenserweiterung *.oct* mit dem ausführbaren Programm von *Octave* verknüpfen. Auf Rechnern mit *UNIX* und *Linux* empfiehlt es sich, einen symbolischen Link, kurz Symlink, etwa durch

```
ln -s /dosc/visualrbfn/bin/visualrbfn.oct /usr/local/bin/vrbfn
```

anzulegen. Dann ersetze man im Folgenden Programmaufrufe von *visualrbfn.oct* durch die des entsprechenden Symlinks, z. B. *vrbfn*.


Tabelle 3.1: Betriebsmodi von *visualrbfn.oct*; bei den Kommandozeilenparametern (KZP) wird nicht auf Groß- und Kleinschreibung geachtet.

KZP	Parameterdatei-Eintrag	Beschreibung
-B	Mode=Mode_Backpropagation	Fehlerrückführungsmodus
-C	Mode=Mode_CrossValidate	Kreuzvalidierungsmodus
-H	Mode=Mode_Help	Hilfemodus
-I	Mode=Mode_Importance	Abschätzung der Wichtigkeit der Prädiktoren (Sensitivitäten)
-L	Mode=Mode_Learn	Lernmodus
-O	Mode=Mode_Optimize	Optimierungsmodus
-V	Mode=Mode_Visualize	Visualisierungsmodus
-W	Mode=Mode_Work	Arbeitsmodus

Ausgabegröße unterstützt. Die Spaltennummern von Eingabegrößen können als Ganzzahlen, Vektoren oder als Bereiche (Datentyp *range*, siehe [Eat97, S. 36 ff.]) angegeben werden.⁸⁾ Kommandozeilenargumente, die Leerzeichen enthalten, sind in Hochkommata ' ' oder „Gänsefüßchen“ " " einzuschließen, z. B. ' [2 : 5 , 8 , 12] '.

Im Lernmodus wird ein RBF-SOM-Netz trainiert und im Arbeitsmodus auf die in einer Musterdatei gespeicherten Datensätze angewendet. Im Fehlerrückführungsmodus findet das Training des RBF-SOM-Netzes durch Gradientenabstieg statt und kann beispielsweise dafür verwendet werden, um die Prädiktoren zu reskalieren oder das Netz ganz oder teilweise überwacht zu trainieren, vgl. S. 30. Der Kreuzvalidierungsmodus dient dazu, um vor allem bei einem kleinen Stichprobenumfang die (globale) Modellgüte besser abschätzen zu können, als dies durch eine einfache Aufteilung in Lern- und Testdaten möglich wäre. Im Hilfemodus werden die Syntax des Programmaufrufs, die Kommandozeilenparameter, die Versionsnummer und der Name des Autors auf die *Standardausgabe* (das ist in der Regel der Bildschirm) ausgegeben. Eine Abschätzung der Wichtigkeit der Prädiktoren, die auch *Sensitivitätsanalyse* genannt wird, geschieht durch Anwenden eines trainierten Modells auf die Daten einer Musterdatei, wobei jeweils ein Prädiktor auf den zugehörigen Mittelwert gesetzt wird und die Änderung des Fehlers, hier der RMSE, Aufschluss über die Wichtigkeit des Prädiktors für das Modell gibt. Im Optimierungsmodus werden die Anzahl von RBF-Zentren sowie die für alle gaußschen Basisfunktionen gleichen Bandbreiten optimiert. Der Visualisierungsmodus generiert Darstellungen zweier- oder dreidimensionaler Graphen von auf \hat{y} basierenden Funktionen, wobei ein oder zwei Prädiktoren variiert und der Rest konstant auf dem jeweiligen Mittelwert oder einem vom Benutzer gewählten Wert gehalten werden.

⁸⁾ Bereiche von Musterdateispalten haben das Format $s1 : s2$ mit ganzzahligen, positiven Spaltennummern $s1$ und $s2$. Dabei ist $s1 \leq s2$. Beide, $s1$ und $s2$, dürfen höchstens der Anzahl von Datenspalten in der Musterdatei entsprechen.

Die Ausführung von *visualrbfn.oct* und den meisten der davon verwendeten Komponenten lässt sich in der Regel durch das gleichzeitige Betätigen der Tastenkombination  vorzeitig abbrechen.

Einzelheiten des Programmablaufs in den einzelnen Betriebsmodi finden sich in dem folgenden Abschnitt.

3.1.4 Programmooptionen und Parameterdateien

Über Kommandozeilenparameter hinaus lassen sich alle Optionen zur Steuerung des Octave-Skriptes *visualrbfn.oct* in *Parameterdateien* wählen. Existiert im aktuellen Arbeitsverzeichnis eine Datei mit dem Namen *options.m*, wird ihr Inhalt unabhängig vom Betriebsmodus gelesen und interpretiert, falls nicht als erster Kommandozeilenparameter der Name einer anderen Parameterdatei angegeben wurde oder das Programm im Hilfe-Modus arbeitet. Werte von Kommandozeilenparametern überschreiben die in Parameterdateien angegebenen Werte.

In einer Klartext-Datei (ASCII-Format) gelten für Zuweisungen nach dem Muster

```
Variablen-Name = Wert;
```

die gleichen Regeln wie für Octave-Skript-Dateien,⁹⁾ siehe [Eat97], z. B.:

```
Mode                = Mode_Work;   # Arbeitsmodus
DataFileName        = "lern.dat";   # Musterdatei
RFBFBandWidthDefault = 0.75;       # Bandbreite
RBFWithLinearTerm   = true;         # RBFN mit
                                                # linearem Term
DataInputColumns    = [1:4, 6];     # Spalten 1-4 und 6
Visu3DConstVector   = zeros(1,5);  # 5-dim. Nullvektor
```

Ein Semikolon nach einer Zuweisung unterdrückt die Ausgabe eines Kommandos auf der Standardausgabe. Kommentare stehen in Zeilen hinter dem Zeichen #.

Die *Octave*-Standarddatentypen für Fließkommazahlen, Matrizen, Bereiche (*range*) sowie Zeichenketten (*string*) sind in [Eat97, S. 29 ff., S. 31 ff., S. 39 ff.] beschrieben. Im Folgenden werden die Bezeichnungen der aus Pascal, C und anderen verbreiteten Programmiersprachen üblichen Standarddatentypen verwendet, z. B. *float* für Fließkommazahl und *integer* für Ganzzahlen. Objekte vom Typ *range* und einzeilige Matrizen werden in diesem Kapitel auch Vektoren (*vector*) genannt. Für boolesche Variable, die nur zwei meistens als „wahr“ oder „falsch“ interpretierte Werte annehmen können, verfügt *Octave* über keinen eigenen Datentyp. Um Fehler bei der Programmierung zu vermeiden und die Lesbarkeit des Programmcodes zu erhöhen, wurden den globalen Variablen *false* (falsch) und *true* (wahr) die Werte 0 bzw. 1 zugewiesen. Die Variablen, welche die Werte von *true* oder *false* annehmen sollen, werden im Folgenden als vom Datentyp

⁹⁾ Wie in den meisten Programmiersprachen üblich sind Punkte statt Kommata Dezimaltrenner für Fließkommazahlen.

bool bezeichnet.¹⁰⁾ Eine Variable vom Aufzählungstyp `enum` besitzt immer genau einen Wert aus einer Liste von Bezeichnern. Auch hier handelt es sich in *visualrbfn.oct* um Variablenbezeichner, denen Ganzzahlen zugewiesen wurden.

Man beachte, dass in *visualrbfn.oct* keine Überprüfung durchgeführt wird, ob die vom Benutzer verwendeten Variablenbezeichner denen von *visualrbfn.oct* entsprechen, ob bei der Wertzuweisung von Variablen diese verträglich zu den Datentypen sind und ob Wertebereiche überschritten werden. In der Regel meldet *Octave* Fehler der letztgenannten Art. Um zu überprüfen, welche Werte den Programmparametern zugewiesen sind und welche durch Kommandozeilenoptionen oder in Parameterdateien vom Benutzer geändert wurden, erzeugt *visualrbfn.oct* die Dateien `optionscheck.txt` und `optionscheck.m`. Ein Beispiel ist auf S. 123 zu finden. Während die erstgenannte Datei ein übersichtliches Protokoll der gewählten Optionen darstellt, kann die zweite als Parameterdatei verwendet werden.¹¹⁾

Die in Parameterdateien benutzten Bezeichner sind ganze oder in üblicher Weise abgekürzte englische Begriffe. Die Schreibweise der in *visualrbfn.oct* verwendeten Bezeichner heben sich von dem *Octave*-Befehlssatz ab, indem sie mit Großbuchstaben beginnen.¹²⁾

Die Optionen beziehen sich auf das eigentliche RBFN, wenn der betreffende Bezeichner mit `RBF` beginnt, auf die selbstorganisierende Karte, wenn er mit `SOM` anfängt und auf die Visualisierung von Netzausgaben als 2D- oder 3D-Grafik, wenn die ersten Buchstaben `visu` lauten. Die Zeichenfolge `BP` am Wortanfang deutet auf eine Option hin, die sich auf die Optimierung der Netzparameter durch Gradientenabstieg mit Fehlerrückführung (Backpropagation) bezieht (vgl. S. 30). Die Einstellungen bezüglich der Musterdatei beginnen mit `Data`. Die Optionen von Kreuzvalidierungen sind durch den Präfix `CV` gekennzeichnet. Der Parzen-Dichteschätzer lässt sich durch Variable mit dem Namensbestandteil `Parzen` beeinflussen.

Ist der Wert von `SOManalyze` größer als null und ist `RBFcentrespos` gleich `RBFcentrespos_SOM`, können Grafik- und Textdateien erzeugt werden, die Informationen über die Daten und die Netzwerk-Ausgaben für die einzelnen Einzugsgebiete \mathcal{X}_k der SOM-Neuronen liefern. Im Lernmodus handelt es sich dabei beispielsweise um die Netzausgaben $\hat{y}(\mathbf{z}_k)$, wenn man die SOM-Kodebuchvektoren als Eingaben für das RBF-

¹⁰⁾ *Octave* vergleicht bei der Auswertung einer `if`-Bedingung, ob der Wert gleich null (falsch) oder ungleich null (wahr) ist [Eat97, S. 75]. Daher lassen sich Ausdrücke wie `if (RBFWithLinearTerm)` mit einer als boolesch bezeichneten Variable `RBFWithLinearTerm` bei der Programmierung verwenden, wobei diese in *Octave* als Fließkommazahl repräsentiert ist.

¹¹⁾ Um beispielsweise bei einem neuen Experiment eine Parameterdatei, etwa `options.m`, zu erzeugen, kann man *visualrbfn.oct* im Lern- oder Arbeitsmodus starten, den Programmablauf abbrechen, die ausgegebene Datei `optionscheck.m` in `options.m` umbenennen und diese dann mit einem Text-Editor modifizieren.

¹²⁾ Die Art und Weise der Vergabe von Namen für Bezeichner ist an die von *Mathematica* angelehnt. Die Wahrscheinlichkeit, dass es im Namensraum von *Octave* zu Konflikten zwischen Bezeichnern der Prozeduren und der globalen Variablen von *visualrbfn.oct* einerseits und andererseits denen von *Octave* kommt, vgl. [Eat97, S. 203], ist als sehr gering einzuschätzen, da die Namen der *Octave*-eigenen Variablen und Funktionen entweder vollständig aus Klein- oder Großbuchstaben (sowie sonstigen Zeichen) bestehen.

SOM-Netzwerk benutzt, die Anzahl von Netzeingaben \mathbf{x}_i , die in ein Einzugsgebiet fallen sowie Minima, Maxima, Mittelwerte und Spannweiten für alle Werte y_i in \mathcal{Y}_k^L . Die lokalen Prognosefehler für die Einzugsgebiete der SOM-Neuronen, die Anzahl der in ein Einzugsgebiet fallenden Netzeingaben und eine Grafik mit den Mustern der Prototypen, die genauso wie das SOM-Gitter angeordnet sind, lassen sich im Arbeitsmodus generieren.

Um bei der Angabe der Streuung der Ausgabevariablen Y innerhalb des Einzugsgebietes eines SOM-Neurons k undefinierte Werte zu kennzeichnen, beispielsweise wenn höchstens ein \mathbf{x}_i der Lerndaten in das betreffende Einzugsgebiet fällt und $\#\mathcal{X}_k^L \leq 1$ gilt, wird der Wert NDEF benutzt, der dann bei grafischen Darstellungen durch eine entsprechende Aufschrift gekennzeichnet wird.

Der Graph der durch das RBFN vermittelten Funktion $\hat{y}: \mathbb{R}^d \rightarrow \mathbb{R}$ lässt sich mit *Visualrbfn* im Visualisierungsmodus partiell veranschaulichen. Eine oder zwei der d Netzeingabe-Größen werden in dem bzw. den in der Musterdatei gegebenen oder durch `Visu3DMinX1` und `Visu3DMaxX1` sowie ggf. `Visu3DMinX2` und `Visu3DMaxX2` gewählten Wertebereich(en) variiert, während die restlichen Eingabekanäle konstant gehalten werden. Fehlt die Angabe eines Vektors (`Visu3DConstVector`) mit den dafür zu verwendenden Konstanten, werden die jeweiligen arithmetischen Mittel benutzt. Ein Wert der ersten bzw. einzigen (nicht konstant gehaltenen) unabhängigen Variablen in der Spalte j_1 wird im Folgenden mit x_{j_1} und der zweiten Größe — falls vorhanden — in der Spalte j_2 mit x_{j_2} bezeichnet. Gilt für die Eingabedimension $d = 1$ oder ist als Musterdatei-Spaltenindex `Visu3DColX2` ein Wert kleiner als eins gewählt, wird ein zweidimensionaler Graph dargestellt. Bei der Verwendung von gefüllten Kreisen zur Markierung von Punkten ist ihr Durchmesser abhängig von der Anzahl der abzubildenden Punkte. Bei mehr als 70 Datenpunkten werden in der Voreinstellung bei Zeitreihendarstellungen nur Linien und keine gefüllten Kreise gezeichnet. Bei dem grafischen Vergleich der Messwerte und Netzausgaben in Streudiagrammen (ohne Zeitachse) weisen die gefüllten Kreise dann nur den halben Durchmesser der sonst verwendeten Kreise auf. Beispiele dafür sind in den Abbildungen 4.4 (S. 72) und 4.19 (S. 90) gegeben.

Vor allem, um bei kleinen Datensätzen die Zuverlässigkeit der Abschätzung der Generalisierungsgüte zu erhöhen, lässt sich eine Kreuzvalidierung durchführen. Die Option `CVReuse` ermöglicht das Wiederverwenden der Ergebnisse vorangegangener langwieriger Kreuzvalidierungen. Weitere Möglichkeiten zur Steuerung dieser Betriebsart finden sich bei den Beschreibungen der entsprechenden Optionen in der Auflistung auf S. 44.

Große Datenmengen können die Ausführungsgeschwindigkeit von *Visualrbfn* beim Lernen und Anwenden deutlich herabsetzen. Vor allem bei den oft wiederholten Auswertungen des Netzwerkes zum Optimieren der Netzwerkparameter durch das Gradientenabstiegsverfahren kann eine *Datenreduktion* von Vorteil sein, die im einfachsten Fall in einer willkürlichen oder zufälligen Auswahl von Daten besteht. Das in *Visualrbfn* implementierte Verfahren fasst dicht beieinander liegende Eingabedatenpunkte \mathbf{x}_{i_q} zusammen, sodass diese nur noch durch ihren Mittelvektor repräsentiert werden. Gleichzeitig werden die Mittelwerte der zugeordneten Werte y_{i_q} der abhängigen Variablen gebildet. Das

Verfahren beginnt mit dem ersten Datenpunkt x_1 und sucht alle Datenpunkte x_{i_1} , deren euklidischer Abstand zu x_1 weniger als `ReduceDataMinDist` beträgt. Diese werden gemittelt, genauso wie ihre zugehörigen Werte y_{i_1} . Dann wird der nächste Datenpunkt in der Datei verwendet, der nicht schon nah genug am ersten liegt und um ihn herum werden ebenfalls alle Punkte gesucht, deren Abstand kleiner als die genannte Schranke ist. Beendet wird das Verfahren, wenn kein Datenpunkt mehr übrig ist. Die Häufigkeiten der einzelnen Elemente in einer solchen Umgebung werden in einer Datei gespeichert. Ignoriert man die darin enthaltenen Informationen, lässt sich der Einfluss von häufigen Mustern auf das nachfolgende Training des Netzwerkes abschwächen.

Unterstützte Methoden zum Positionieren der m RBF-Zentren sind die zufällige Auswahl von Vektoren, deren Komponenten Werte im gleichen Wertebereich wie die entsprechenden Trainingsdaten besitzen (`RBFCentres_Random`), die Selektion der ersten m Eingabevektoren x_i im Trainingsdatensatz (`RBFCentres_Samples`), die Durchführung eines K-Means-Clusterverfahrens (`RBFCentres_KMeans`) sowie die Anwendung einer SOM (`RBFCentresPos_SOM`, vgl. S. 17).

Bei der Verwendung einer SOM zum Positionieren der RBF-Zentren kann man die Breite und die Höhe des SOM-Gitters durch die Optionen `SOMHeight` und `SOMWidth` wählen, was z. B. zur Konstruktion von Ketten ($m \times 1$) dient. Wird hingegen nur die Anzahl von RBF-Zentren (`RBFCentresNum`) angegeben, versucht `visualrbfn.oct` zunächst ein quadratisches Gitter zu bilden und dann schrittweise die Anzahl von Neuronen in der Höhe zu erniedrigen und in der Breite zu erhöhen, bis die gewünschte Anzahl von Zellen annähernd erreicht ist. Die voreingestellte Anzahl von 24 SOM-Neuronen liefert dabei beispielsweise ein SOM-Gitter mit 6×4 Zellen. Um eine schon angepasste SOM ohne erneutes Training zum Positionieren von RBF-Zentren zu verwenden, lässt sich die boolesche Variable `SOMReuse` einsetzen. Wird `RBFCentresNum = 0` (und `RBFWithLinearTerm = true`) gesetzt, simuliert `visualrbfn.oct` ein LNN.

In der Version 4.5 unterstützt `visualrbfn.oct` neben gaußschen Basisfunktionen auch die Thin-Plate-Spline-Funktion (vgl. Gl. 2.4 auf S. 17). Wird der Wert `Gauss` für `RBFType` nicht verwendet, ist das Programm weder im Optimierungs- noch im Fehlerrückführungsmodus (vgl. Tab. 3.1) ausführbar.

Die Bandbreiten h_k der einzelnen radialen Basisfunktionen des RBFNs können voneinander verschieden sein. Bei der Bandbreitenbestimmung über die nächsten Nachbarn wird die Bandbreite einer RBF gleich dem maximalen euklidischen Abstand der `RBFBandwidthNN` nächstliegenden RBF-Zentren gesetzt.

Für Berechnungen einer Likelihood mit Parzen-Dichteschätzern kann die Bandbreite der Gauß-Glocken im Unterschied zu der in [Bis94] genannten Heuristik durch den Benutzer angegeben werden. Der voreingestellte Wert 0 bewirkt, dass die mittlere Bandbreite des schon trainierten RBFNs (siehe Datei `l_rbf_bandwidths.dat`) verwendet wird. Der Name der Datei mit Lerndaten für den Dichteschätzer kann mit dem der Datei übereinstimmen, die dem Training des RBFNs zugrunde liegt.

In der folgenden Auflistung der in einer Parameterdatei wählbaren Optionen steht links jeweils in einer eigenen Zeile der Bezeichner der beschriebenen Option, ihr Datentyp

und der voreingestellte Wert; rechts daneben befindet sich eine kurze, stichwortartige Beschreibung.

BPContinue bool true	Fortsetzung der Optimierung aller Netzwerk-Gewichte durch Gradientenabstieg (Backpropagation)
BPetaS float 0.001	Schrittweite η_s bei der Veränderung der Gewichte der Sensitivitätsneuronen beim Gradientenabstieg, siehe Gl. 2.16 (S. 31)
BPetaW float 0.001	Schrittweite η_w bei der Veränderung der Gewichte der Ausgabeschicht beim Gradientenabstieg, vgl. BPetaS
BPetaZ float 0.001	Schrittweite η_z bei der Veränderung der RBF-Zentren beim Gradientenabstieg, vgl. BPetaS
BPLogStepNumber integer 3	Anzahl von Schritten beim Gradientenabstieg, nach denen jeweils die Protokoll- und Netzparameter-Dateien geschrieben werden
BPMomentumB float 0.8	Momentum-Term für das Gradientenabstiegsverfahren, siehe Gl. 2.16 (S. 31)
BPNumSteps integer 25	Anzahl von Schritten beim Gradientenabstieg, nach der das Verfahren abbricht
BPStopRMSE float 0.005	Schranke für den RMSE, bei deren Unterschreiten das Gradientenabstiegsverfahren abbricht
BPSVD bool true	Schalter zur Durchführung einer Singulärwertzerlegung (SVD) nach dem Ende des Gradientenabstiegsverfahrens zur weiteren Optimierung der Gewichte des linearen Ausgabeneurons
CVCrossValidate bool true	Steuerung, ob eine Kreuzvalidierung (<code>true</code>) oder eine einfache Validierung durchgeführt wird (<code>false</code>)
CVNumValSamples integer 5	Anzahl von maximal bei einer Kreuzvalidierung verwendeten Validierungsdaten; siehe CVRatioValSamples

CVRandomOrder bool true	Schalter, ob die Auswahl der Trainings- und Validierungsdaten bei einer Kreuzvalidierung zufällig erfolgt
CVRatioValSamples float 0	Anteil der Daten in der aktuellen Musterdatei, der maximal bei einer Kreuzvalidierung zur Validierung verwendet wird; nur wenn der Wert gleich 0 ist, wird CVNumValSamples verwendet
CVReuse bool false	Schalter für die Wiederverwendung der Resultate einer früheren Kreuzvalidierung
CVShowProgress bool true	Fortschrittsanzeige bei einer Kreuzvalidierung, falls der Wert gleich true ist
DataFileName string learn.dat	Name der Musterdatei
DataInputColumns integer vector 0	Vektor oder Bereich von Indizes der Spalten (kleinster möglicher Index: 1) in der Musterdatei, die als Netzeingabe dienen sollen; die Voreinstellung 0 erzwingt, dass bei $d + 1$ Spalten in der Lerndatenmatrix die ersten d Spalten als Eingabe und die letzte als Ausgabe betrachtet werden
DataOutputColumns integer vector 0	Spaltenindizes der Musterdatei, die als Netzausgaben dienen sollen, siehe DataInputColumns; viele Funktionen des Simulationsprogramms sind in der Version 4.5 nur für eine Ausgabedimension programmiert
DataRescaleInput bool false	Schalter, ob die Datei bp_rbf_s.dat (falls sie vorher erzeugt wurde) für die automatische Reskalierung der Eingabegrößen ausgewertet wird
Mode enum Mode_Learn	Modus des Programmablaufs (vgl. Tab. 3.1 auf S. 39)
NDEF integer 12345	Kennzeichen für nicht definierte, fehlerhafte oder fehlende Werte ^(e) in der Musterdatei

ParzenBandWidth float 0	Bandbreite h für die Basisfunktionen des Parzen-Dichteschätzers; der Wert 0 bewirkt, dass die mittlere Bandbreite der RBFs verwendet wird; siehe auch ParzenDensity
ParzenDataFile string learn.dat	Name der Datei, welche die Lerndaten für den Parzen-Dichteschätzer enthält; in der Regel ist das die Datei mit den Lerndaten für das RBFN; siehe ParzenDensity
ParzenDensity bool false	Schalter für die Berechnung einer Likelihood mit dem in Gl. 2.11 (S. 22) dargestellten Parzen-Dichteschätzer
PlotActivities bool false	Aktivitäten der RBF-Neuronen und die Abstände der Eingabedatenvektoren von den RBF-Zentren werden im Arbeitsmodus auf der SOM-Gitter-Anordnung als EPS-Grafiken ausgegeben (wenn zuvor eine SOM trainiert wurde)
PlotGraphics bool true	Schalter, ob Diagramme der Werte y_i und \hat{y}_i , der Residuen sowie des SOM-Quantisierungsfehlers in Abhängigkeit von den Nummern der Daten sowie ein (\hat{y}_i, y_i) -Streudiagramm und ein Histogramm der Residuen erzeugt und in EPS-Dateien gespeichert werden
Quiet bool false	Unterdrückung von Programmausgaben auf der Standardausgabe; die Option überschreibt den Wert von VerboseLevel und setzt ihn auf 0
RBFBandWidthConst bool false	Schalter, ob die Bandbreiten für alle RBF-Neuronen konstant (gleich RBFBandWidthDefault) sein sollen (true) oder nach dem Nächste-Nachbarn-Verfahren bestimmt werden (false)
RBFBandWidthDefault float 0.4	Voreinstellung der konstanten Bandbreite für alle radialen Basisfunktionen im Netzwerk
RBFBandWidthNN integer 8	Anzahl von Nachbar-Zentren, die beim Nächste-Nachbarn-Verfahren zur Berechnung der Bandbreiten des RBFNs herangezogen werden

RBFBandWidthStretch float 1.0	Faktor, mit dem alle Bandbreiten des RBFNs multipliziert werden (nur in Verbindung mit nicht-konstanten Bandbreiten sinnvoll; sonst verändere man RBFBandWidthDefault)
RBFCalcLearnError bool true	Berechnung des RMSEs für die Anwendung des Netzes auf Lerndaten im Lernmodus (Zeitersparnis, falls gleich false)
RBFCentresNum integer 24	Anzahl von RBF-Zentren (entspricht der Anzahl von SOM-Neuronen bei der Verwendung einer SOM)
RBFCentresPos enum RBFCentresPos_SOM	Methode zum Positionieren der RBF-Zentren: RBFCentresPos_Random (1), RBFCentresPos_Samples (2), RBFCentresPos_KMeans (3) und RBFCentresPos_SOM (4)
RBFPESupport bool false	Schalter für die Berechnung der Fehlerbalken, wie in Unterabschnitt 2.2.1 (S. 20) beschrieben
RBFType string Gauss	Name der verwendeten RBF; neben der gaußschen Basisfunktion (Gauss) kann durch TPSpline auch die <i>Thin-Plate-Spline-Funktion</i> nach Gl. 2.4 (S. 17) gewählt werden
RBFVISupport bool false	Schalter für die Berechnung der in Unterabschnitt 2.2.2 (S. 20) genannten Größen des VI-Netzes; die Aktivität der einzelnen RBF-Neuronen wird zudem protokolliert, im Lernmodus wird die Dichte der Eingabedaten geschätzt, falls gilt: RBFVISupport=true; die Verwendung von linearen Neuronen außerhalb der Ausgabeschicht verhindert die Unterstützung von RBFVISupport
RBFWithBias bool true	Schalter für die Verwendung eines absoluten Gliedes (Bias-Element) w_0 in Gl. 2.2 (S. 17); es gilt $w_0 = 0$, falls der Wert false gewählt wird
RBFWithLinearTerm bool false	Erweiterung des RBFNs um einen linearen Term gemäß Gl. 2.5 (S. 18)

ReduceData	Datenreduktion, siehe S. 42
bool	
false	
ReduceDataMinDist	Mindestabstand zwischen Eingabedaten bei der Datenreduktion, falls ReduceData=true
float	
0.05	
SOMAnalyze	Analyse der SOM-Ausgaben, siehe S. 41
bool	
true	
SOMCBVFileName	Name der Datei, in der die Kodebuchvektoren beim SOM-Training gespeichert sind
string	
l_som_cbv.smf	
SOMHeight	Anzahl von Neuronen als Höhe des rechteckigen SOM-Gitters; der Wert 0 erzwingt eine automatische Berechnung von Höhe und Breite des Gitters gemäß der Anzahl (RBFCentresNum) der Kodebuchvektoren (siehe S. 43)
integer	
0	
SOMNumTrainingSteps	Anzahl von Schritten beim Training der SOM, vgl. [KHKL96]
integer	
100000	
SOMReuse	Benutzung einer schon trainierten SOM im Lernmodus des RBFNs ohne ein erneutes Training der SOM
bool	
false	
SOMWidth	Anzahl von Neuronen als Breite des rechteckigen SOM-Gitters, vgl. SOMHeight
integer	
0	
VerboseLevel	Schwelle für die Ausgaben des Programms auf die Standardausgabe; der Wert 0 bedeutet dabei, dass keine Ausgaben erfolgen, bei einem VerboseLevel von 3 werden alle Statusmeldungen und Hinweise ausgegeben
integer	
3	
Visu3DColX1	Spaltenindex für die Musterdatei mit Werten der Variablen X_{j_1}
integer	
1	

<p>Visu3DColX2 integer 2</p>	<p>Spaltenindex für die Musterdatei mit Werten der Variablen X_{j_2}</p>
<p>Visu3DConstVector float vector <i>Mittelwerte</i></p>	<p>Vektor mit jeweils einem Wert für jeden der d Prädiktoren X_j ($j = 1, \dots, d$), den X_j annehmen soll, falls er bei der Visualisierung des Graphen von \hat{y} nicht variiert wird</p>
<p>Visu3DMaxX1 float 0</p>	<p>Maximum der Werte x_{j_1} bei der Visualisierung des Graphen von \hat{y}; das Gleichsetzen der Werte von Visu3DMinX1 und Visu3DMaxX1 erzwingt, dass der in der Musterdatei gegebene Wertebereich von X_{j_1} ausgeschöpft wird (Voreinstellung)</p>
<p>Visu3DMaxX2 float 0</p>	<p>Maximum der Werte x_{j_2} im Visualisierungsmodus, vgl. Visu3DMaxX1</p>
<p>Visu3DMeshLabelX1 string " "</p>	<p>Achsenbeschriftung für die x_{j_1}-Achse im Visualisierungsmodus; die Voreinstellung bedeutet, dass keine Beschriftung erscheint</p>
<p>Visu3DMeshLabelX2 string " "</p>	<p>Achsenbeschriftung für die x_{j_2}-Achse im Visualisierungsmodus, vgl. Visu3DMeshLabelX1</p>
<p>Visu3DMeshLabelY string " "</p>	<p>Achsenbeschriftung für die \hat{y}-Achse im Visualisierungsmodus, vgl. Visu3DMeshLabelX1</p>
<p>Visu3DMinX1 float 0</p>	<p>Minimum der Werte x_{j_1} im Visualisierungsmodus, vgl. Visu3DMaxX1</p>
<p>Visu3DMinX2 float 0</p>	<p>Minimum der Werte x_{j_2} im Visualisierungsmodus, vgl. Visu3DMaxX1</p>
<p>Visu3DNumGridPoints integer 15</p>	<p>Anzahl s von Stützstellen (Gitterpunkten) entlang der x_{j_1}- und — falls vorhanden — x_{j_2}-Achse im Visualisierungsmodus; bei zwei unabhängigen Größen werden die Funktionswerte zur Darstellung eines dreidimensionalen Graphen an s^2 vielen Stellen berechnet</p>

3.1.5 Modifizierbare Shell- und Octave-Skripten

Eine Reihe von Shell- und Octave-Skripten kann aus dem Verzeichnis, in dem sich auch *visualrbfn.oct* befindet, in das jeweilige Arbeitsverzeichnis kopiert und dort mit einem Text-Editor verändert werden:

`callvisusom.sh`

Das Skript ruft das Visualisierungswerkzeug *visusom* (siehe S. 62) auf, um spezielle grafische Darstellungen von Aktivitäten der Neuronen oder Netzausgaben auf der Basis der verwendeten SOM zu erzeugen, z. B. um nur bestimmte Testdaten darzustellen.

`cbv_array.sh`

Linien- oder Balkendiagramme der Komponenten der Kodebuchvektoren von SOMs mit rechteckiger Anordnung lassen sich dem Aufbau des SOM-Gitters entsprechend darstellen. Beispiele finden sich in Abb. 4.11 (S. 86) und Abb. 4.12 (S. 87). Die zu übergebenden Kommandozeilenparameter sind der Name der Datei, welche die Kodebuchvektoren enthält, die der Breite der SOM entsprechende Spaltenzahl in der Grafikausgabe, die darzustellenden Spalten aus der Datei der Kodebuchvektoren, der Dateinamensstamm der Ausgabedateien im EPS-Format und schließlich die Angabe, ob durch die Voreinstellung `lines` ein Linien- oder durch `bars` ein Balkendiagramm erzeugt wird.

`label_cases.sh`

Das Skript erzeugt für die Eingabedatenobjekte Bezeichnungen, die bei der Darstellung der SOM-Neuronen als Beschriftungen benutzt werden; beispielsweise kommen bei Zeitreihen Nummern (Voreinstellung) oder das Datum und allgemein eine Klassen-Zugehörigkeit in Frage. Der erste, zwingend zu übergebende Kommandozeilenparameter beinhaltet den Dateinamen der Lerndaten. Als zweiter Parameter kann der Name einer Datei angegeben werden, in der die Ausgaben von `label_cases.sh` gespeichert werden (Voreinstellung: `labeleddata.tmp`).

`label_hits.sh`

Sollen die Anzahlen von Testmustern, die in die Einzugsgebiete der SOM-Neuronen fallen, in der SOM-Darstellung erscheinen, wird dieses Skript ausgeführt. Damit lässt sich ablesen, wie oft ein Neuron Sieger beim Anlegen der Testdaten war. Als Kommandozeilenparameter ist dem Skript der Name der Datei mit den Kodebuchvektoren zu übergeben.

`mesh3doptions.oct`

Spezielle Optionen, die in dieser Datei im Octave-Format [Eat97, S. 125 ff.] angegeben sind, dienen der Anpassung der Darstellung der zwei- oder dreidi-

mensionalen Graphen im Visualisierungsmodus.¹³⁾

`train_som.sh`

Dieses Skript initialisiert eine SOM, steuert ihr Training und ruft zur Darstellung der Resultate als U-Matrix- und Sammon-Abbildung die entsprechenden Programme aus der Simulationssoftware für SOM-Netzwerke (*SOM_PAK*, siehe Unterabschnitt 3.2.3 ab S. 62) auf. An das Skript werden vier optionale Parameter übergeben: die Breite und die Höhe des SOM-Gitters, jeweils angegeben in Anzahlen von Neuronen, die Anzahl der Iterationen beim Training sowie der Name der Datei mit den Lerndaten. Möglicherweise sind in der Datei der Pfad (`sompakpath`) zu den ausführbaren Dateien von *SOM_PAK* oder die Anzahl von Iterationen bei der Erzeugung der Sammon-Abbildung (`sammon_iterations`) anzupassen.

3.1.6 Ausgaben

Die Simulationssoftware *visualrbfn.oct* generiert Status-, Warn- und Fehlermeldungen in englischer Sprache an der Standardausgabe, deren Umfang durch die Optionen `VerboseLevel` und `Quiet` beeinflusst werden kann.¹⁴⁾ Beispielsweise wird die in Sekunden gemessene Laufzeit von *visualrbfn.oct* am Ende einer Programmausführung nur dann zu Protokoll gegeben, wenn der Wert von `VerboseLevel` mindestens gleich 2 ist.

Darüber hinaus werden auch eine ganze Reihe verschiedener Text- und Grafikdateien erzeugt. Der Dateinamenssuffix `.eps` weist dabei auf das Standardformat *Encapsulated PostScript* (EPS) hin. Die Anzahl der erzeugten EPS-Dateien kann sehr groß sein, da die Grafiken je nach Darstellung für alle Dimensionen oder alle Testmuster und zudem mit absoluten und relativen Qualitätswerten entsprechenden Graustufen erzeugt werden.

Bei den gespeicherten Textdateien bedeutet die Dateinamenserweiterung `.dat`, dass es sich um Dateien in dem in Unterabschnitt 3.1.2 auf S. 37 beschriebenen Datenformat handelt. Die Endung `.smf` wird hingegen für Dateien vergeben, die Daten im Format von *SOM_PAK*-Map-Files [KHKL96, S. 11] enthalten, z. B. für Kodebuch-Vektoren und Ausgaben der *SOM_PAK*-Module. Der Qualifier `.txt` wird für andere Textdateien verwendet. Temporäre Dateien besitzen den Suffix `.tmp`. Das bei einer Kreuzvalidierung angefertigte Protokoll heißt `crossvalidation.prt`.

Die Dateinamen bestehen nur aus Kleinbuchstaben, um eine übersichtliche alphabetische Sortierung der Verzeichnis-Inhalte zu gewährleisten. Außerdem wurde auf Leer- und Sonderzeichen in den Dateinamen verzichtet. Die vor dem ersten Unterstrich stehenden Buchstaben des Ausgabedateinamens geben den Betriebsmodus von *visualrbfn.oct* an, in dem die Datei generiert wurde. Dabei steht `bp` für den Fehlerrückführungs-

¹³⁾ Um beispielsweise den angezeigten Wertebereich der abhängigen Größe dem Intervall [1.5:6] gleichzusetzen, fügt man in die Datei `mesh3doptions.oct` in eine eigene Zeile `gset zrange [1.5:6]` ein.

¹⁴⁾ Es ist nicht möglich, den Umfang der Ausgaben der von *visualrbfn.oct* aufgerufenen Programme zu steuern.

cv für den Kreuzvalidierungs-, l für den Lern-, v für den Visualisierungs- und w für den Arbeitsmodus. Dann folgen, durch einen Unterstrich getrennt, eine Abkürzung des Namens der erzeugenden Netzkomponente von RBFSOM (*som*, *rbf*, *rbfsom*) oder ein Hinweis auf Daten (*data*) und schließlich weitere Abkürzungen, welche den Inhalt der jeweiligen Dateien genauer spezifizieren. Die Inhalte der Zeilen der meisten der genannten Dateien beziehen sich auf die Datensätze in den Musterdatei-Zeilen mit den gleichen Zeilennummern.

Im Folgenden sind die Inhalte der von *Visualrbfn* erzeugten ASCII-Text-Dateien stichwortartig beschrieben. Je nach den gewählten Optionen bei der Programmausführung werden nicht alle der genannten Dateien erzeugt. EPS-Dateien mit einem gleichlautenden Dateinamensstamm enthalten jeweils entsprechende Informationen in grafischer Darstellung.

`bp_data_rescaled.dat`

Daten der Musterdatei, die durch s_j ($j = 1, \dots, d$) reskaliert wurden

`bp_rbf_out.dat`

Netzwerkprognosen, Daten der abhängigen Variablen und die Differenzen zwischen beiden nach dem Gradientenabstieg

`bp_rbf_out_start.dat`

wie `bp_rbf_out.dat`, allerdings nach der Initialisierung und vor dem Gradientenabstieg

`bp_rbf_rmse.dat`

RMSE für jede Iteration während des Gradientenabstiegs (kein Dateikopf mit der Angabe der Spaltenzahl)

`bp_rbf_s.dat`

Skalierungsfaktoren (Sensitivitäten) s_j ($j = 1, \dots, d$) für den Wert x_j der Prädiktorenkomponente j , die durch das Gradientenabstiegsverfahren mit Fehler-rückführung gefunden wurden

`bp_rbf_steps.dat`

Anzahl der durchgeführten Schritte beim Gradientenabstiegsverfahren

`bp_rbf_tmp_D.dat`

Momentaufnahmen der beim Gradientenabstiegsverfahren erzeugten Matrizen, wobei der Platzhalter D die Werte Z für RBF-Zentren, S für Skalierungsfaktoren und W für die Gewichte des linearen Ausgabeneurons annehmen kann

`bp_rbf_w.dat`

Gewichte des linearen Ausgabeneurons nach beendetem Gradientenabstiegsverfahren und anschließender (falls `BPSVD=true`) Singulärwertzerlegung

`bp_rbf_w_end.dat`

Gewichte des linearen Ausgabeneurons (nach beendetem Gradientenabstiegsverfahren)

`cv_rbf_out.dat`

Ausgaben des RBFNs für die Validierungsdaten bei der Kreuzvalidierung, aufgelistet in der gleichen Reihenfolge wie die Lerndaten

`crossvalidation.prt`

Protokoll einer Kreuzvalidierung mit Gütemaßen und statistischen Kennwerten (Minimum, Maximum, Mittelwert, Standardabweichung)

`l_rbf_activities.dat`

Aktivitäten der RBF-Neuronen für die Lerndaten

`l_rbf_b.dat`

Gewichte des linearen Ausgabeneurons des RBFNs

`l_rbf_b_pe.dat`

Gewichte des linearen Ausgabeneurons des RBFNs beim Lernen von erwarteten Fehlern, allerdings nur für die RBF-Zentren ohne den linearen Teil oder ein Bias-Neuron, falls Fehlerbalken ermittelt werden, vgl. Unterabschnitt 2.2.1 auf S. 20

`l_rbf_bandwidths.dat`

Bandbreiten h_k der RBFs

`l_rbf_centres.dat`

RBF-Zentren \mathbf{z}_k ($k = 1, \dots, m$)

`l_rbf_vi_conflimax.dat`

Vorstufe der Konfidenzgrenzen bei dem VI-Netz-Ansatz für Lerndaten (vgl. Unterabschnitt 2.2.2 auf S. 20)

`l_rbf_vi_density.dat`

Hilfsgrößen für die Dichteschätzungen des VI-Netzes

`l_rbf_vi_predlimaux.dat`

Vorstufe der Prognoseintervallgrenzen bei dem VI-Netz-Ansatz für Lerndaten (vgl. Unterabschnitt 2.2.2 auf S. 20)

`l_rbf_som_cbv_response.dat`

Netzausgaben $\hat{y}(\mathbf{z}_k)$ für in das Netzwerk eingegebene Kodebuchvektoren \mathbf{z}_k ($k = 1, \dots, m$)

`l_rbf_som_cbv_response_rel.dat`

[0,1]-skalierte Werte aus `l_rbf_som_cbv_response.dat`

`l_rbf_som_cbv_vi_cl.dat`

(halbe) Konfidenzintervallbreiten des VI-Netzes bei den SOM-Kodebuchvektoren als Eingaben; die Dateinamensstammendungen `abs` und `rel` bei abgeleiteten EPS-Dateien deuten an, dass die Graustufen die Daten im Bereich

zwischen 0 und Maximum bzw. zwischen Minimum und Maximum darstellen, wobei der kleinste Wert jeweils schwarz und der größte weiß wiedergegeben wird

`l_rbf_som_cbv_vi_dens.dat`

durch das VI-Netz ausgegebene Likelihoods der Dichteschätzungen für die SOM-Kodebuchvektoren; siehe `l_rbf_som_cbv_vi_cl.dat`

`l_rbf_som_cbv_vi_pl.dat`

(halbe) Prognoseintervallbreiten des VI-Netzes bei den SOM-Kodebuchvektoren als Eingaben; siehe `l_rbf_som_cbv_vi_cl.dat`

`l_rbf_out.dat`

Netzwerkprognosen $\hat{y}(\mathbf{x}_i)$, Messwerte y_i und deren Differenzen für die Lerndaten in drei Spalten

`l_som_data.dat`

Lerndaten-Matrix zum Trainieren der SOM (entspricht der Musterdatei)

`l_som_density_reltotal.dat`

für jedes SOM-Neuron k Dichteschätzung als Quotient zwischen $\#\mathcal{X}_k^L$ (Anzahl von Lerndaten \mathbf{x}_i im Einzugsgebiet des SOM-Neurons k) und der Gesamtzahl der Lerndaten

`l_som_density_relmax.dat`

Quotient zwischen $\#\mathcal{X}_k^L$ und $\max\{\#\mathcal{X}_\ell^L \mid \ell = 1, \dots, m\}$ für jedes SOM-Neuron k

`l_som_dim.dat`

Breite und Höhe der SOM (angeordnet in einer Zeile mit zwei Spalten)

`l_som_hits.dat`

$\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$ für alle Lerndaten (Anzahl von Lern-Eingabedaten im Einzugsgebiet des SOM-Neurons, dessen Kodebuchvektor dem aktuellen Datenpunkt am nächsten liegt); in der entsprechenden Grafikdatei sind die den SOM-Neuronen zugeordneten Kästchen mit der genannten Anzahl beschriftet, der Grauton des Kästchen-Hintergrundes liegt zwischen Schwarz für den Wert 0 und Weiß für die maximale Anzahl

`l_som_idx.dat`

Nummern k der SOM-Neuronen und Angabe ihrer Lage im zweidimensionalen Gitter (der erste Spalten- und Zeilenindex ist jeweils 0)

`l_som_y_max.dat`

$\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ für alle Lerndaten

`l_som_y_mean.dat`

arithmetisches Mittel $\bar{y}_{\kappa(\mathbf{x}_i)}$ aller y_i in $\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L$ (Lerndaten)

- `l_som_y_min.dat`
min($\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L$) für alle Lerndaten
- `l_som_y_range.dat`
Spannweite $\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L) - \min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ für alle Lerndaten
- `l_som_y_range_rel.dat`
an der maximalen Spannweite relativierte Werte aus der Datei `l_som_y_range.dat`
- `optionscheck.m`
Bezeichner und Werte der bei einer Simulation verwendeten Optionen zur Programmsteuerung; kann über das Überprüfen und die Dokumentation der verwendeten Programmoptionen hinaus auch als Vorlage für benutzerspezifische Parameterdateien dienen, siehe S. 41 sowie das Beispiel auf S. 123
- `optionscheck.txt`
vgl. `optionscheck.m`, allerdings anders formatiert (keine Parameterdatei)
- `v_rbf_2d.dat`
Ausgabe zweidimensionaler Punkte des Graphen einer partiellen Netzausgabefunktion zur Weiterverarbeitung in anderen Grafikprogrammen (Visualisierungsmodus)
- `v_rbf_2d_data.dat`
Eingabedaten, die einer 2D-Visualisierung der partiellen Netzausgabefunktion zu Grunde liegen; in einer weiteren Spalte rechts sind Zufallszahlen zwischen null und eins als Pseudo-Messwerte für Y eingefügt; die Datei kann in einem weiteren Schritt im Arbeitsmodus des Netzes ausgewertet werden, um z. B. die verschiedenen Likelihoods der Parzen- und VI-Netz-Dichteschätzer und Schätzungen der Konfidenz- und Prognoseintervalle zu erzeugen und dann in einem weiteren Schritt mit *Gnuplot* oder anderer Software grafisch darzustellen
- `v_rbf_3d.dat`
Ausgabe dreidimensionaler Punkte (in drei Spalten angeordnete Zahlen) des Graphen einer partiellen Netzausgabefunktion zur Weiterverarbeitung in anderen Grafikprogrammen (Visualisierungsmodus)
- `w_rbf_activities.dat`
Aktivitäten $\phi_k(\mathbf{x}_i)$ der RBF-Neuronen für die Testdaten
- `w_rbf_activities_rel.dat`
Verhältnis der Aktivitäten der RBF-Neuronen für die Testdaten und der maximalen Aktivität
- `w_rbf_dist_xz.dat`
Matrix mit den an ihrem Maximum relativierten euklidischen Abständen aller Testdaten \mathbf{x}_i von den RBF-Zentren \mathbf{z}_k

w_rbf_errorbars.dat

Fehlerbalken nach dem Predictive-Error-Bars-Ansatz für die Testdaten

w_rbf_vi.dat

Ausgaben des VI-Netzes im Arbeitsmodus (siehe S. 20)

w_rbf_out.dat

Netzwerkprognosen $\hat{y}(x_i)$, Messwerte y_i und deren Differenzen für die Testdaten in drei Spalten

w_rbf_som_rmse.dat

(lokaler) RMSE im Einzugsgebiet der zugehörigen SOM-Neuronen für alle Testdaten

w_rbf_som_rmse_rel.dat

an dem maximalen RMSE der SOM-Einzugsgebiete relativierte Werte aus w_rbf_som_rmse.dat

w_som_data.dat

Matrix der Testdaten zur Anwendung der SOM

w_som_hits.dat

Anzahl von Testdaten, die in das Einzugsgebiet eines jeden SOM-Neurons fallen (im Unterschied zu l_som_hits.dat)

w_som_hits_rel.dat

an der maximalen Anzahl von Elementen innerhalb eines Einzugsgebietes relativierte Daten aus w_rbf_som_hits.dat

Grafikdateien, die kein Pendant als Textdatei besitzen, sind:

cbv_array_abs.eps

Linien- oder Balkendiagramme der Komponenten der SOM-Kodebuchvektoren in der Anordnung des SOM-Gitters als Ausgabe von cbv_array.sh, z. B. Abb. 4.11 (S. 86)

cbv_array_rel.eps

Darstellung wie in cbv_array_abs.eps, wobei die einzelnen Vektor-komponenten der Kodebuchvektoren auf das Intervall [0,1] abgebildet sind; ist beispielsweise der Ordinatenwert des (von links gezählt) k ten Punktes in einem Teildiagramm gleich 0, hat die k te Komponente des dem Diagramm zugehörigen Kodebuchvektors einen bezüglich der Kodebuchvektoren minimalen Wert, z. B. Abb. 4.12 (S. 87)

l_som_cbv_pj.eps

Graustufendarstellungen über dem SOM-Gitter der Werte der Kodebuchvektorkomponenten (S. 16) mit den Nummern j ($j = 1, \dots, d$), z. B. Abb. 4.10 (S. 86)

`l_som_cbv_tr.eps`

Trajektorien als durch einen Polygonzug verbundene Grafikelemente, die den Siegerneuronen der SOM (mit den ähnlichsten Prototypen) für eine Folge von Eingabedatenvektoren entsprechen, z. B. Abb. 4.13 (S. 87)

`l_som_sammon_sa.eps`

Sammon-Abbildung der SOM-Kodebuchvektoren, z. B. Abb. 4.2 (S. 70) und Abb. 4.9 (S. 85)

`l_som_umatrix_average.eps`

U-Matrix-Darstellung der SOM, z. B. Abb. 4.8 (S. 84)

`w_rbf_out_lines.eps`

Darstellung der Messwerte y_i in der Musterdatei und der Netzausgaben $\hat{y}(x_i)$ im Arbeitsmodus über den Nummern der Daten, beginnend mit eins („Zeitreihendarstellung“), z. B. Abb. 4.4 (S. 72) und Abb. 4.19 (S. 90)

`w_rbf_out_xy.eps`

Darstellung der Messwerte y_i in der Musterdatei und der Netzausgaben \hat{y}_i im Arbeitsmodus als Punkte (\hat{y}_i, y_i) („Streudiagramm“), z. B. Abb. 4.19 (S. 90)

Die Klartext-Datei `work_results.txt` enthält wichtige Ergebnisse der Anwendung des RBFSOM-Netzes im Arbeitsmodus sowie weitere Informationen über die Netzwerksimulation: Datum, Uhrzeit, Arbeitsverzeichnis, Musterdateiname, Anzahl der Datensätze, Fehlerquadratsumme, mittlere Fehlerquadratsumme, deren Wurzel und Bestimmtheitsmaß. Die Netzwerkausgaben und Qualitätsmaße für einen Testdatensatz sind in Tab. 3.3 zu finden. Je nach den gewählten Einstellungen für den Programmablauf, z. B. falls `RBFPESupport=false` oder `RBFSVSupport=false` gilt, sind einige der genannten Merkmale in der Datei nicht enthalten.

Für die Merkmale in den einzelnen Spalten der Datei lassen sich deskriptiv-statistische Maße berechnen, beispielsweise Minima, Maxima, Mittelwerte und Standardabweichungen. Dafür bieten sich *Octave*, *Excel*, *Mathematica*, *Matlab* sowie *Awk* an.

3.1.7 Laufzeit und weitere Hinweise

Die Dauer der Ausführung von *Visualrbfn* und seinen Komponenten hängt von vielen Faktoren ab. Eine wesentliche Rolle spielen die Anzahl und die Komplexität der zu lernenden oder zu testenden Daten, die Anzahl von RBF-Zentren und die Methode ihrer Positionierung sowie die verwendete Computerhardware und das Betriebssystem.

Um die *Laufzeit* einschätzen zu können, wurden Lern- und Testdaten aus je 1'000 Datensätzen und 8 Spalten zusammengestellt (wie sie in Abschnitt 4.2 ab S. 80 beschrieben sind). Zum Trainieren eines RBFSOM-Netzwerkes mit 7 Netzeingaben und einer Ausgabegröße, einer 13×6 -SOM, einem Kreuzvalidierungsanteil von 50%, ohne Fehlerrückführung und mit den sonstigen Standardeinstellungen sowie allen Auswertungsmöglichkeiten (Grafiken und Zuverlässigkeitsmaße) benötigte ein Personal-Computer mit einem Intel-Pentium-4-Prozessor mit einer Taktfrequenz von 2.66 GHz, 256 MB RAM, *Windows XP*

Tabelle 3.3: Bedeutung des Inhaltes der Datei `work_results.txt`. Erläuterungen zu den Größen, die sich auf das VI-Netz beziehen, sind auf S. 20 zu finden.

Spaltenkopf	Bedeutung der Werte
No.	fortlaufende Nummer i des Datensatzes (\mathbf{x}_i, y_i) in der aktuellen Musterdatei (beginnend mit 1), auf die sich die jeweilige Zeile bezieht
Net-output	Netzausgabe (Prognose) \hat{y}_i
Data	Messwert y_i
Resid.	Residuum (Differenz zwischen Messwert und Netzausgabe): $y_i - \hat{y}_i$
MaxAct.	maximale Erregung der RBF-Neuronen: $\max\{\phi_k(\mathbf{x}_i) \mid k = 1, \dots, m\}$
Winner	Index k des SOM-Neurons mit dem \mathbf{x}_i nächstliegenden Kodebuchvektor \mathbf{z}_k , d. h. $\kappa(\mathbf{x}_i)$
Mindist.	euklidischer Abstand von \mathbf{x}_i zum nächsten SOM-Kodebuchvektor \mathbf{z}_k
SOMSupport	Anzahl von Lerndaten, die dem entsprechenden Siegerneuron (vgl. Spalte „Winner“) zugeordnet sind: $\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$
Dispers.	Spannweite als Differenz zwischen den Werten der Spalten „YMaxSOM“ und „YMinSOM“
YMinSOM	$\min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$
YMaxSOM	$\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$
MeanCls.	arithmetisches Mittel über die Elemente von $\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L$
VIMaxAct.	entspricht dem Inhalt der Spalte „MaxAct.“ (VI-Netz)
VIAveAct.	mittlere Aktivierung der RBF-Neuronen (VI-Netz)
VIDensity	Likelihood (Dichteschätzung durch das VI-Netz)
VIconfLim.	Konfidenzintervallbreite (VI-Netz)
VIPredLim.	Prognoseintervallbreite (VI-Netz)
PEB	Prognose des Fehlers (Predictive-Error-Bars)
ParzDens.	Likelihood (Parzen-Dichteschätzer)

und *Cygwin* 2 Minuten und 52 Sekunden. Dabei wurden 75 Dateien (darunter 29 EPS-Dateien) neu erzeugt, die einschließlich der 141 KByte für die Dateien mit Lern- und Testdaten 1'823 KByte Platz auf der Festplatte belegten. Im Arbeitsmodus waren 6 Minuten und 5 Sekunden nötig, um alle Auswertungen durchzuführen und in Ausgabedateien zu speichern. Einschließlich der im Lernmodus erzeugten Dateien generierte die Software 3'098 Dateien, von denen allerdings allein 3'036 auf EPS-Dateien entfielen. Der benötigte Plattenplatz betrug 13.34 MB, davon belegten die Grafikdateien 9.17 MB.

Die Laufzeit für das Lernen auf einem Rechner mit dem Betriebssystem *Microsoft Win-*

dows 2000 und *Cygwin*, einem 800-MHz-Intel-IV-Prozessor und 128 MB RAM betrug 4 Minuten. Die Arbeitsphase dauerte etwas mehr als 13 Minuten. Dieser Rechner benötigte bei der Verwendung des Betriebssystems *SuSE-Linux* (Version 7.0 für i386, Kernel 2.2.16) mit 8 Minuten und 13 Sekunden mehr als doppelt so viel Zeit für das Lernen und mit über 17 Minuten für das Auswerten der Testdaten ebenfalls wesentlich länger als unter *Windows 2000*.

Ohne Erzeugung der meisten Grafiken und ohne Berechnung von Zuverlässigkeitsmaßen dauerte das Lernen des RBF-SOM-Netzwerkes mit dem schnelleren Rechner unter *Windows XP* 32 Sekunden. Das Testen nahm 78 Sekunden in Anspruch.

Die Dauer für das Erzeugen einer Grafik im Visualisierungsmodus liegt im Bereich von wenigen Sekunden. Bei dem Gradientenabstiegsverfahren kann die Programmausführung jedoch viele Stunden dauern.

Um auf der Standardausgabe erscheinende Meldungen zu protokollieren, empfiehlt sich ihre Umlenkung durch `>` bzw. `>>` in eine Datei, z. B.

```
visualrbfn.oct -l lern.dat >protokoll.log
```

Die mit *Gnuplot* erzeugten Abbildungen lassen sich — z. B. für die Angabe der Achsenbeschriftungen in einer anderen Sprache — ändern, indem man die Steuerbefehle enthaltende Datei `gnuplotcmd.tmp` mit einem Text-Editor modifiziert und

```
gnuplot gnuplotcmd.tmp
```

in der Kommandozeile ausführen lässt.¹⁵⁾

3.2 Technische Aspekte

Dieser Abschnitt enthält wichtige technische Informationen zu dem Programmpaket *Visualrbfn*, die für eine Installation, Anpassung und Erweiterung der Software benötigt werden. Die Kenntnis des vorangegangenen Abschnittes wird vorausgesetzt.

Die Simulationssoftware *Visualrbfn* wird im Rahmen der „GNU General Public License“, wie sie von der „Free Software Foundation“ veröffentlicht wurde, kostenlos weitergegeben.

Weitere Hinweise sind als Kommentare im Quellcode von *visualrbfn.oct* zu finden.

3.2.1 Systemvoraussetzungen und Installation

Die Hardwarevoraussetzungen für die Installation und die Anwendung von *Visualrbfn* sind gering. Die in der Distribution enthaltenen Dateien belegen weniger als 0.5 MB Platz auf einer Festplatte. In Unterabschnitt 3.1.7 (S. 57) finden sich Angaben zu Personal-Computern, mit denen *Visualrbfn* getestet wurde.

¹⁵⁾ Je nach verwendeter Programmversion von *Gnuplot* kann der Befehl unter *MS-Windows* auch

```
pgnuplot.exe gnuplotcmd.tmp
```

lauten.

Die nachfolgend genannte Software ist für die Nutzung aller Möglichkeiten, die *Visualrbfn* bietet, erforderlich:

- ▷ *Bourne-Shell* oder *BASH*
- ▷ unter *UNIX*, *Linux* sowie *Cygwin* vorhandene Befehle (bzw. Programme) `head`, `tail`, `cut`, `paste` und `awk`
- ▷ *Octave*, siehe Unterabschnitt 3.2.2 (S. 61)
- ▷ *SOM_PAK*, siehe Unterabschnitt 3.2.3 (S. 62)
- ▷ *visusom*, siehe Unterabschnitt 3.2.3 (S. 62)
- ▷ *Gnuplot*
- ▷ *Ploticus*

Gnuplot und *Ploticus* sind frei verfügbare, über die Kommandozeile steuerbare Grafikprogramme, welche die Speicherung von Grafikdateien im EPS-Format ermöglichen und u. a. für *UNIX*, *Linux* und *MS-Windows* verfügbar sind. Eine Steuerdatei für *Ploticus* wird durch das in *Visualrbfn* enthaltene *Awk*-Skript `plotbars` des Verfassers generiert. Für *MS-Windows* empfiehlt sich die Installation von *Cygwin*, eine von der Firma *Red Hat* entwickelte *UNIX*-Umgebung, die eine Shell-Programmierung ermöglicht und die viele der unter *UNIX* sowie *Linux* bekannten Werkzeuge bereitstellt. Hinweise zur Installation der genannten Software entnehmen man den entsprechenden Hilfe- und „Readme“-Dateien sowie den Benutzerhandbüchern.

Alle Dateien in den Distributionsdateien von *Visualrbfn*, vgl. Abschnitt 6.6 (S. 128), lassen sich mit Hilfe zahlreicher Programme entpacken¹⁶⁾ und unterhalb des aktuellen Arbeitsverzeichnisses in die vorgegebenen Unterverzeichnisse ablegen. Weitere Informationen sind in der Datei `Readme` enthalten, die zu der Distribution von *Visualrbfn* gehört. Die *Verzeichnisstruktur*, in der das Softwarepaket *Visualrbfn* installiert wird, besteht aus einem Hauptverzeichnis, z. B. `visualrbfn`, und den darin enthaltenen Unterverzeichnissen `bin` für ausführbare Shell-Skripten und *Octave*-Funktionsdateien sowie `examples` mit Beispielen für die Verwendung von *Visualrbfn*.

Beim Setzen der *Pfade* beachte man, dass der Platzhalter für das aktuelle Verzeichnis vor den Verzeichnissen angegeben wird, in denen andere ausführbare Dateien von *Visualrbfn* gespeichert sind. Dadurch werden anstelle der vorgegebenen Shell-Skripten aus dem genannten `bin`-Verzeichnis gleichnamige, vom Benutzer angepasste Dateien im Arbeitsverzeichnis verarbeitet (vgl. Unterabschnitt 3.1.5, ab S. 50).¹⁷⁾

¹⁶⁾ Entsprechende Befehle sind beispielsweise `unzip vrbfn.zip` oder `tar xvfz vrbfn.tgz`

¹⁷⁾ Ein Beispiel für das Setzen des Pfades mit dem aktuellen Verzeichnis an erster Stelle:

```
export PATH=./visualrbfn/bin:$PATH
```

Aufgrund von Sicherheitsbedenken gilt dies jedoch nicht für den Benutzer „root“.

3.2.2 Programmierung des Netzwerksimulators in Octave

Der zentrale Baustein der Software zur Simulation des RBF-SOM-Netzwerkes, die Datei `visualrbfn.oct`, wurde vom Verfasser mit *GNU-Octave* programmiert. *Octave* ist eine Hochsprache^(e), die hauptsächlich für numerische Berechnungen entworfen wurde. Sie besitzt eine einfach zu bedienende Kommandozeilen-Schnittstelle, um lineare und nicht-lineare Probleme numerisch zu lösen und numerische Experimente mit einer Sprache durchzuführen, die weitgehend mit jener von *Matlab* kompatibel ist. Die Software kann überdies zur Stapelverarbeitung und Programmierung eingesetzt werden [Eat97, S. 5]. *Octave* ist u. a. für die 32-Bit-Versionen von *MS-Windows* sowie für *Linux* und *AIX* verfügbar und kann frei im Rahmen der Vereinbarungen der „GNU General Public License“ vertrieben werden. Die Sprache ist leicht durch benutzerdefinierte Funktionen erweiterbar. Weiterhin lassen sich dynamisch geladene Module einbinden, die beispielsweise in C, C++, Fortran oder anderen Sprachen geschrieben sind [Eat02].

In den dieser Arbeit zugrunde liegenden Experimenten wurden die *Octave*-Versionen 2.0.13, 2.0.16 und 2.1.31 eingesetzt.

Da von *Octave* kein Referenzaufruf^(e) von Funktionsparametern unterstützt wird [Eat97, S. 63], wurden viel Speicher benötigende Matrizen und einige andere Variable global deklariert. Einige Prozeduren und Funktionen sind in externen Dateien ausgelagert. Der Gebrauch solcher „function files“ [Eat97, S. 92 ff.] trägt nicht nur zur Kompaktheit der Skriptdateien und der Übersichtlichkeit der Programm-Module bei, sondern ermöglicht es auch, die Module in andere *Octave*-Skripten einzubinden.¹⁸⁾

Die erste Zeile von `visualrbfn.oct` lautet:

```
#!/usr/bin/octave -qf
```

Ist die ausführbare Datei `octave` nicht im Verzeichnis `/usr/bin` zu finden, ist stattdessen das entsprechende Verzeichnis anzugeben oder in `/usr/bin` ein symbolischer Link auf das Programm anzulegen.¹⁹⁾ Gegebenenfalls sind die Pfadangaben in den ersten Zeilen auch in den anderen Skriptdateien anzupassen, die sich im selben Verzeichnis wie die Datei `visualrbfn.oct` befinden.

Anstelle der von *Octave* bereitgestellten Funktionen `error` und `usage` wurden selbstgeschriebene Funktionen `Error` und `Help` verwendet. Die meisten Ausgaben des Programms auf die Standardausgabe erfolgen nur, wenn der betreffende zugewiesene Wert mindestens gleich dem der benutzerspezifischen Einstellung von `VerboseLevel` ist.

Die Erweiterung von `visualrbfn.oct` durch weitere radiale Basisfunktionen, wie sie beispielsweise in [Zel94, S. 229] zu finden sind, geschieht nach den Vorgaben durch die Funktionen `Gauss(x, w, h)` und `TPSpline(x, w, h)`. Da *Octave* keine Zeiger^(e) auf Funktionen unterstützt, wie es beispielsweise von der Sprache C bekannt ist, werden die Namen der Funktionen durch den `feval`-Mechanismus [Eat97, S. 73 f.] ausgewertet.

¹⁸⁾ Die Skript-Dateien für *Octave* tragen normalerweise die Endung `.oct` [Eat97, S. 94], während für Funktionsdateien empfohlen wird, die Dateinamenserweiterung `.m` zu verwenden [Eat97, S. 92 ff.].

¹⁹⁾ Das optionale Argument `-q` in der ersten Zeile von `visualrbfn.oct` unterdrückt Meldungen beim Starten von *Octave* und `-f` verhindert das Verarbeiten von Initialisierungsdateien.

Octave benutzt *Gnuplot* zur Erzeugung grafischer Darstellungen. Um die Funktionalität zu erweitern, mussten an einigen Stellen in *visualrbfn.oct* Steuerdateien für *Gnuplot* generiert werden.

Um eine Weiterentwicklung der Software von internationalen Programmierern zu erleichtern, sind die Kommentare in englischer Sprache verfasst.

3.2.3 Erweiterung der SOM_PAK-Software

Bei der Simulation von SOM-Netzwerken hat sich das von Teuvo KOHONEN und seinen Mitarbeitern entwickelte Programmpaket *SOM_PAK* bewährt. Der Quellcode steht über das Internet zur Verfügung und muss mit einem C-Compiler übersetzt werden. Dadurch lassen sich ausführbare Programme für viele Betriebssysteme erzeugen, darunter *Linux*, *UNIX* und *MS-Windows* mit *Cygwin*. Einzelheiten dazu sowie Hinweise zum Setzen der Pfade entnehme man [KHKL96].

Von dem Software-Paket *SOM_PAK* werden aus *visualrbfn.oct* sowie dem Bourne-Shell-Skript *train_som.sh* die Programme²⁰⁾ *randinit*, *visual*, *qerror*, *planes*, *umat*, *vsom* und *sammon* aufgerufen.

Die Musterdateien für *Visualrbfn* können von *SOM_PAK* gelesen und verarbeitet werden. Hingegen unterstützt *Visualrbfn* nicht alle Möglichkeiten, die das Dateiformat für *SOM_PAK* [KHKL96, S. 11 ff.] bietet.²¹⁾

Der Verfasser erweiterte *SOM_PAK* im Rahmen der vorliegenden Dissertation um das ebenfalls in der Programmiersprache C geschriebene Werkzeug *visusom*, das die grafische Darstellung von Werten bei der Anwendung des RBF-SOM-Netzes auf dem SOM-Gitter und damit der verdeckten RBF-Schicht ermöglicht. Bei den in EPS-Dateien gespeicherten Abbildungen des SOM-Gitters entsprechen helle Graustufen großen Werten für die darzustellende Größe, während kleine Zahlen dunkleren Tönen zugeordnet sind. Insbesondere deuten weiße Felder auf Werte von eins und darüber hin, während null und negative Werte durch schwarze Bereiche dargestellt werden. Allerdings besteht über die Option *-negative* die Möglichkeit, „Negativdarstellungen“ mit invertierten Graustufen zu erzeugen.

Das Programm wird von *visualrbfn.oct* gestartet, lässt sich aber auch (beispielsweise zur Modifikation von automatisch erzeugten Grafiken) einzeln ausführen.

Die folgende Auflistung der Kommandozeilenparameter von *visusom* enthält links, jeweils untereinander angeordnet, den Bezeichner des zu beschreibenden Parameters, seinen Datentyp (nur *string* für Zeichenketten und *integer* für Ganzzahlen) und bei optionalen Parametern den voreingestellten Wert sowie rechts daneben eine kurze Beschreibung.

²⁰⁾ Bei *MS-Windows*-Programmen ist an die genannten Bezeichner der Dateinamenssuffix *.exe* angehängt.

²¹⁾ Während *SOM_PAK* beispielsweise die Markierung von fehlenden Werten durch das Symbol *x* ermöglicht, wird dies von *Visualrbfn* nicht unterstützt.

Erforderliche Kommandozeilenparameter:

-cin string	Datei mit Kodebuchvektoren und Angaben über das SOM-Gitter
-din string	Musterdatei mit zu visualisierenden Daten

Optionale Kommandozeilenparameter:

-buffer integer 0	gepuffertes Lesen von Daten
-case integer 0	Nummer (Zeile in der Datenmatrix) des zu visualisierenden Musters in der Datei; der Wert 0 bedeutet, dass für alle Muster in der Eingabedatei eine Grafikdatei erzeugt wird
-cnuminfname integer 1	Erscheinen der Nummer (-case) des zu visualisierenden Musters im Dateinamen hinter dem ersten Teil des Dateinamensstammes, siehe -epsbasename
-epsbasename string	erster Teil des Dateinamensstammes der EPS-Ausgabedateien; Voreinstellung ist der Dateinamensstamm der Musterdatei
-framed integer 0	Erzeugung einer schwarzen Umrahmung um die Grafik
-help	Anzeige eines Hilfetextes mit Informationen über die Programmversion und den Autor
-labels integer 1	Anzeige der Beschriftungen für die SOM-Zellen in der Grafik, wenn diese in der Eingabedatei in der letzten Spalte (hinter den Datenspalten) vorhanden sind (1 bedeutet „ja“, 0 heißt „nein“)
-ndef integer 12345	Indikator für nicht definierte Werte in der Eingabedatei
-ndefstr string ndef	Beschriftung der SOM-Elemente einer Grafik, für die der betreffende Wert nicht definiert ist
-negative integer 0	Invertierung der Graustufen in den Grafiken, falls der Wert größer als 0 ist

-ps	Erzeugung der Datei im PostScript- statt im EPS-Format; Ausgabe- dateien besitzen dann den Dateinamenssuffix .ps
integer	
0	
-squares	Darstellung von Quadraten anstelle von Kreisen für die SOM- Neuronen, wenn der Wert größer als 0 ist
integer	
0	

Die Kommandozeile

```
visusom -cin l_som_cbv.smf -din w_rbf_activities.dat
        -epsbasename w_rbf_act -case 4
        -squares 1 -negative 1 -framed 1
```

erzeugt beispielsweise unter Verwendung der in der Datei `l_som_cbv.smf` enthaltenen Informationen über das SOM-Gitter (und die Kodebuchvektoren) und den in der Datei `w_rbf_activities.dat` gespeicherten Aktivitäten der RBF-Neuronen die EPS-Datei `w_rbf_act_00004.eps` und durch Quadrate symbolisierte SOM-Neuronen. Die Graustufen werden invertiert, sodass große Werte dunkel und kleine Werte hell dargestellt sind. Außerdem ist die Grafik durch eine schwarze Linie umrahmt. Beispiele für mit *visusom* erzeugte Grafiken finden sich in Abb. 4.7 (S. 75) sowie in Abschnitt 4.2 (S. 80 ff.).

3.3 Diskussion

Das Programmpaket *Visualrbfn* wurde in erster Linie zur Computersimulation des RBF-SOM-Modells entwickelt. Überdies kann es aber auch in verschiedenen Modi als RBFN mit zwei Basisfunktionstypen und vier (mit Fehlerrückführung sogar fünf) Verfahren zur Positionierung der RBF-Zentren oder als Lineares Regressionsmodell betrieben werden. Dies ermöglicht einen Vergleich der Generalisierungsfähigkeit unterschiedlicher Modelle, zu denen auch dasjenige gezählt werden kann, das aus den mittleren Werten der Ausgabegröße innerhalb der SOM-Klassen besteht. Von den sonstigen Erweiterungen sind die Fehlerrückführung und der Gradientenabstieg zum vollständig überwachten Training und zur günstigen Reskalierung der Eingabegrößen, die Abschätzung der Wichtigkeit von Prädiktoren, die Kreuzvalidierung, der Parzen-Dichteschätzer, das VI-Netzwerk, die Fehlerbalkenschätzung und die zahlreichen Visualisierungsmöglichkeiten der verschiedenen Netzausgaben und Zuverlässigkeitsmaße besonders hervorzuheben.

Die Verwendung von *Octave* für die wesentlichen Module ermöglichte durch seine einfache Syntax und die umfassenden numerischen Funktionen und Datenstrukturen (u. a. Matrizen) sowie die grafischen Möglichkeiten von *Gnuplot* ein schnelles Umsetzen des Modells in eine Simulationssoftware. Im Vergleich dazu hätte eine Implementierung als *EDL*-Klient in die Experimentalumgebung N^2E^4 [Klö94] einen größeren Arbeits- und Zeitaufwand erfordert. Die Ausführungsgeschwindigkeit des Programms ist bei den im

Rahmen dieser Arbeit benutzten Datensätzen und Anwendungen nicht entscheidend, zumal die Laufzeit im Sekunden- bis Minutenbereich lag. Gleichwohl kann sich das Gesamtschrittverfahren des Gradientenabstiegs einschließlich der wiederholten Fehlerberechnung bei größeren Datensätzen als zeitaufwendig erweisen. Es gilt abzuwägen, ob sich eine Übersetzung der Software oder einzelner Komponenten in C, C++ oder andere Compilersprachen auszahlen würde.

Die Benutzung vieler verschiedener Programme anstatt eines einzigen oder weniger Werkzeuge bedeutet zunächst einen Mehraufwand bei der Installation und kann sich als problematisch erweisen, wenn einzelne Teile nicht installiert sind. Allerdings wird dies durch eine schnellere und einfachere Entwicklung der Software und den Einsatz von sich als zuverlässig erwiesenen Werkzeugen ausgeglichen, die viele Optionen bieten.²²⁾ Die Verwendung von ausschließlich kostenloser Software, die für wissenschaftliche Zwecke frei verfügbar ist oder der „General Public License“ unterliegt, die überdies u. a. unter den gängigsten Betriebssystemen lauffähig ist, erleichtern eine weite und schnelle Verbreitung der Software. Der modulare Aufbau des Programmpaketes gestattet das Ersetzen einzelner Komponenten durch modifizierte Bausteine. Beispielsweise kann die Visualisierung von Netzwerkausgaben als 3D-Gitter-Darstellungen anstelle von *Gnuplot* auch mit *Mathematica* oder anderen Grafikprogrammen erfolgen und das Trainieren und Anwenden der SOM lässt sich auch mit einer anderen Software als mit *SOM_PAK* durchführen.

Durch das Anpassen von Shell-Skripten im Arbeitsverzeichnis können mit geringem Aufwand benutzerspezifische Anpassungen vorgenommen werden, ohne das Hauptprogramm der Software modifizieren oder die vorgegebenen Skripten für alle Anwender und Anwendungen ändern zu müssen. Dieser Ansatz wurde durch das Verwenden virtueller Funktionen, die Möglichkeit des Überladens in der objektorientierten Programmierung sowie sogenannte „User exits“ (SAP-Software) inspiriert.

Die von *Visualrbfn* unterstützten Klartext-Dateien bieten gegenüber Binärdateien einige Vorteile. Beispielsweise sind sie leicht zu erstellen, mit zahlreichen Werkzeugen (u. a. durch *UNIX*-Befehle *awk*, *cut*, *head*, *tail*) zu bearbeiten und in nahezu beliebigen Text-Editoren sowie *Excel* einfach les- und modifizierbar. Ausgabedateien können mit *Excel*, *Mathematica*, *Gnuplot* oder anderer Software grafisch dargestellt oder statistisch ausgewertet werden. Die von den aus *Visualrbfn* aufgerufenen Programmen erzeugten Grafikdateien im EPS-Format lassen sich in Dokumente von Satzsystemen wie $\text{T}_{\text{E}}\text{X}/\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ sowie von Textverarbeitungssoftware wie *Word für Windows* einbinden.²³⁾ Im Vergleich zu Pixel-Grafiken benötigen die darin gespeicherten Vektorgrafikobjekte in der Regel wenig Speicherplatz, lassen sich gut skalieren und liefern in vielen Fällen eine überragende Qualität im Druck, die nur von den Möglichkeiten (z. B. der Auflösung) des Ausgabe-

²²⁾ Viele der benötigten Komponenten sind üblicherweise in *UNIX*- und *Linux*-Distributionen sowie in *Cygwin* enthalten.

²³⁾ Um die EPS-Dateien am Computer-Bildschirm zu betrachten, in andere Formate zu konvertieren oder auf nicht PostScript-fähigen Druckern auszugeben, können *GhostView* bzw. *GhostScript* verwendet werden. Mehr Informationen und eine Bezugsquelle finden sich im Internet unter <http://www.cs.wisc.edu/~ghost/>.

gerätes abhängt. Bei der Visualisierung der verdeckten Schicht der RBF-Neuronen spielt das neu entwickelte Programm *visusom* neben den in *SOM_PAK* enthaltenen Werkzeugen eine wesentliche Rolle.

Die Software *Visualrbfn* besitzt keine grafische Benutzerschnittstelle^(e) (GUI). Der Ansatz über die Kommandozeilensteuerung sowie über Shell-Skripten bietet für den als im Umgang mit Computern erfahrenen Anwender eine schnelle und einfache Möglichkeit der Durchführung von Computersimulationen des RBFSOM-Netzwerkes. Einzelne Versuchsparameter können mit einem beliebigen Text-Editor in Parameterdateien schnell verändert werden. Gleichzeitig sind die Versuchsbedingungen dadurch weitgehend dokumentiert. Über Shell-Skripten lassen sich Simulationsexperimente automatisieren.

Schließlich kann die Erstellung einer benutzerfreundlichen und ansprechend gestalteten GUI einen erheblichen Mehraufwand bedeuten, der für die Entwicklung und Anwendung des Netzwerkmodells zunächst nicht gerechtfertigt erscheint. Bewährt sich die Software weiterhin nach der nun weitgehend abgeschlossenen Entwicklungs- und ersten Erprobungsphase, kann eine GUI für verschiedene Windows-Systeme beispielsweise mit *Java* (*Swing*) oder *Tcl/Tk* erstellt werden. Denkbar ist aber auch die Integration des RBFSOM-Modells in andere, bewährte Softwarepakete, etwa *N²E⁴* oder *Trajan*.

4 Fallstudien aus der Limnologie

Die in diesem Kapitel dargestellten Fallstudien zeigen Ergebnisse von Anwendungen des RBFSOM-Netzwerkes und anderer Netztypen in der *Limnologie*, der Lehre von den Binnengewässern.¹⁾ Dabei beschäftigt sich der Abschnitt 4.1 zunächst mit der limno-ökologischen Fragestellung nach der Beschreibung und Prognose der Häufigkeit von Wasserinsekten in Abhängigkeit von abiotischen Umweltfaktoren, die auf verhältnismäßig wenigen, multivariaten Daten basieren. Die Möglichkeiten, aber auch die Schwierigkeiten und ihre Lösungsansätze bei der Anwendung des RBFSOM-Netzwerkes zur Bildung eines Wetter-Abfluss-Modells für einen Bach und sein *Einzugsgebiet* auf der Grundlage eines umfangreichen Datensatzes werden in dem Abschnitt 4.2 vorgestellt.

4.1 Prognose der Abundanz von Wasserinsekten

4.1.1 Einführung

Jeder Organismus und jede Population von Lebewesen ist von den in ihrem Lebensraum wirkenden Umweltfaktoren und den gegebenen Ressourcen abhängig. Nur innerhalb der ökologischen Nische, in die sich jede Spezies im Laufe ihrer Evolution eingepasst hat, können die Individuen einer Art wachsen und sich reproduzieren. Eine fundamentale Nische wird durch biotische Faktoren wie beispielsweise intra- und interspezifische Konkurrenz, Räuber (Prädation), Parasitismus und Krankheiten aufgrund von Viren, Bakterien und Pilzen eingeschränkt, sodass man in der Natur nur die realisierten Nischen einer Art erkennen kann. Liegen Erkenntnisse über diese und die wesentlichen Umweltfaktoren und Ressourcen an einem Ort vor, lassen sich Aussagen über die dortigen Möglichkeiten der Entwicklung einer Population machen, vgl. [Rem92].

Bei der Vorhersage der *Abundanz*^(e) einer Art, d. h. die Anzahl von Organismen bezogen auf eine Flächen- oder Raumeinheit [ST83, S. 8], aufgrund von als bekannt vorausgesetzten Umweltbedingungen sind einige Schwierigkeiten zu erwarten. Eine Art kann aus vielerlei Gründen in einem Biotop fehlen, der ihre Nische bietet und die Populationsdichte nimmt aus vielerlei Gründen keineswegs immer ihren theoretischen Maximalwert an. Überdies können Probleme, u. a. bei der Messung und der Auswahl relevanter Umweltfaktoren, der Wahl einer repräsentativen Sammelmethode sowie der Bestimmung einiger

¹⁾ Limnologie: zu griechisch *limne* „Teich, Landsee“ und *logos* „Kunde, Lehre“ [Wah86, S. 838]; nach [Bro94, Band 13 (Lah-Maf), S. 404] untersucht sie die physikalischen, chemischen und ökologischen Eigenschaften der Gewässer, ihre fisch- und wasserwirtschaftliche Nutzung, ihre Belastung mit Abwässern sowie die Abwasserreinigung und Wasseraufbereitung.

Taxa auftreten. Daher scheint es einfacher zu sein, die Präsenz einer Art an einem Ort auszuschließen als ihr Vorkommen vorherzusagen oder sogar ihre Abundanz zu schätzen. Trotzdem gelang schon 1954 die Beschreibung der Populationsdichte von Blasenfüßen der Art *Thrips imiginis* aufgrund meteorologischer Einflüsse mit Hilfe eines linearen, multiplen Regressionsmodells [AB54, BHT91, S. 572]. In jüngerer Zeit wurden nicht-lineare KNNs u. a. bei der Schätzung der Häufigkeiten von Fließgewässer bewohnenden Fischen wie Bachforellen (*Salmo trutta fario*) [LBB⁺96, BLDB96] und Elritzen (*Phoxinus phoxinus*) [MLD97] sowie der Abundanz von Makrozoobenthos-Taxa [Oba98] eingesetzt.

Zeitliche Schwankungen der Umweltfaktoren und das Entwicklungsstadium der Organismen, in denen unterschiedliche Nischen besetzt werden können, müssen bei der Modellierung der Populationsdynamik berücksichtigt werden. In früheren Studien, z. B. [BSW⁺97, Dap98, SBW⁺99, WDS00], wurde ein „sliding window“-Ansatz benutzt, bei dem ein gleitendes Zeitfenster die Werte der Variablen für die letzten Zeitpunkte und -räume beinhaltet, um die monatlichen Abundanzen von Arten mindestens zeitweise im Wasser lebender Insekten sowie anderer wirbelloser Tiere zu beschreiben. In der vorliegenden Untersuchung dienten hingegen feste, den Lebenszyklus der Taxa überdeckende Zeitfenster zur Prognose der jährlichen Anzahlen von ausgewählten Eintags-, Stein- und Köcherfliegenarten in Abhängigkeit von abiotischen Umweltfaktoren und der Häufigkeit der Eltern- generation. Dabei kamen einige der in der vorliegenden Arbeit vorgestellten Zuverlässigkeitsmaße und Möglichkeiten der Veranschaulichungen zum Einsatz.

Einzelne Ergebnisse der vorliegenden Fallstudie sind in [OWWS01] zu finden. Die Problemstellungen motivierten die Weiterentwicklung der Methoden und der Software *Visualrbrn*.

4.1.2 Material und Methoden

Die Datenerhebung erfolgte am *Breitenbach*, einem intensiv erforschten kleinen Bach in der Nähe der hessischen Stadt Schlitz (50°40' Nord, 9°45' Ost). Die wichtigste und fortwährend Wasser führende Quelle liegt in einer Höhe von etwa 310 m ü.N.N. etwa auf der Hälfte der insgesamt ca. 4 km Gesamtlänge des Breitenbaches. Die Mündung in die Fulda befindet sich ca. 220 m über dem Meeresspiegel. Der Bachboden besteht aus Buntsandstein-Kies unterschiedlicher Körnung, größeren Steinen, Sand, Schlamm, Ton und Detritus. Im Oberlauf rinnt der Bach entlang des Waldrandes, an dem vorwiegend Kiefern (*Pinus sp.*), Rotbuchen (*Fagus sylvatica*), Stiel-Eichen (*Quercus robur*) und Hainbuchen (*Carpinus betulus*) stehen. Bachabwärts ist das Gewässer hauptsächlich von Wiesen und Weiden gesäumt. Detaillierte Beschreibungen des Baches einschließlich botanischer Aspekte der Ufervegetation finden sich in [Cox90] und [Rin74]. Eine topographische Karte sowie Informationen zu physikalischen und chemischen Eigenschaften sind in [WSM94] wiedergegeben.

Der mittlere monatliche Niederschlag betrug im Untersuchungszeitraum von 1969 bis 1998 an der Mess-Station „Haus B“ etwa 55 ℓ/m^2 (min. 4.9 ℓ/m^2 , max. 181 ℓ/m^2), die mittlere monatliche Wassertemperatur lag bei 11.6 °C (min. 3 °C, max. 25 °C) und der

Mittelwert der Monatsmaxima des Abflusses wurde mit 40 ℓ/s (0.8–616 ℓ/s) gemessen. Die dieser Fallstudie zugrunde liegenden Daten über die Lebensgemeinschaft bestanden aus jährlichen Individuenzahlen der 40 häufigsten von über 100 im Breitenbach im Untersuchungszeitraum gefundenen Insektenarten aus den Ordnungen der Eintags-, Stein- und Köcherfliegen (Ephemeroptera, Plecoptera und Trichoptera, kurz EPT), die in der Emergenzfalle [MZ99] „Haus B“ auf 6 m Bachlänge gefangen wurden, siehe Abb. 4.1, Abb. 6.1 (S. 122) und Tab. 6.1 (S. 121). Die Falle bedeckte abhängig vom Abfluss eine Wasserfläche von etwa 5 m^2 . Die Variablen zur Beschreibung der abiotischen Umweltbedingungen bestanden aus den monatlichen Maxima der Wassertemperatur und des Abflusses sowie der monatlichen Regenmenge in dem ca. 10 km^2 überdeckenden Einzugsgebiet des Breitenbaches über einen Zeitraum von 13 Monaten, in denen sich die meisten der untersuchten Tiere vom Ei bis zur flugfähigen Imago (bzw. Subimago) entwickelten.

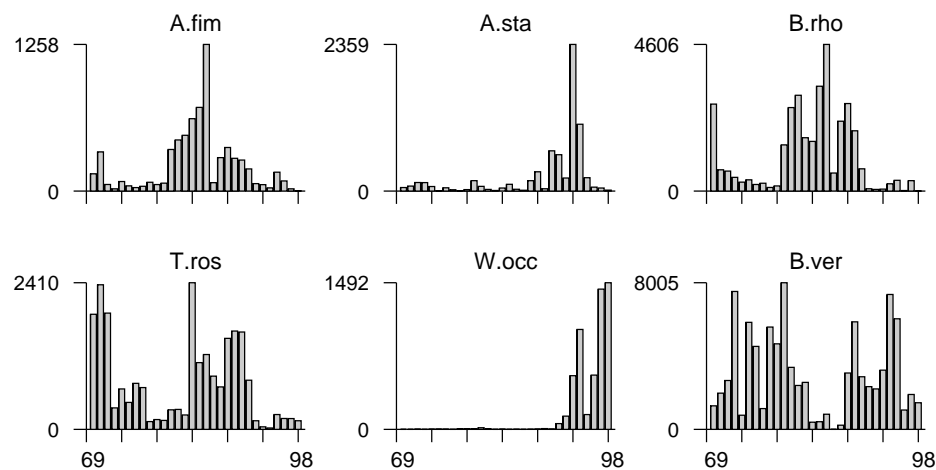


Abbildung 4.1: Jährliche Abundanzen ausgewählter EPT-Arten in der Emergenzfalle „Haus B“ im Untersuchungszeitraum 1969–1998; die Abkürzungen der Art-namen sind in Tab. 6.1 (S. 121) erklärt.

Die jährlichen Anzahlen der in der Falle gefangenen Individuen der Insektenarten wurden relativiert, indem die Werte durch das jeweilige Maximum im Untersuchungszeitraum dividiert wurden. Für ein Modell wurden die Abundanzen der Art *Apatania fimbriata* zusätzlich logarithmisch skaliert. Den Einfluss extremer Abfluss-Messwerte dämpfte ebenfalls eine logarithmische Skala. Alle anderen Größen waren [0,1]-skaliert.

Die Testdaten zur Beurteilung der Generalisierungsfähigkeit der KNNs bestanden aus Messergebnissen der zufällig ausgewählten Jahre 1974, 1982, 1984, 1988, 1992 und 1996. Die Daten der übrigen 24 Jahre wurden für das Training sowie teilweise bei Kreuzvalidierungen zur Auswahl relevanter Eingabegrößen und zur Optimierung von Netzwerkparametern verwendet.

Eine Datenexploration wurde mit SOMs durchgeführt. GRNNs fanden mit einer schrittweisen Methode und Genetischen Algorithmen (vgl. S. 36) relevante Prädiktoren für die Approximation der Regressionsfunktionen mit RBFSOM-Netzen.

4.1.3 Ergebnisse

Die Ähnlichkeiten der Muster relativer, jährlicher Abundanzen von 40 EPT-Arten in der Emergenzfalle „Haus B“ über 30 Jahre, vgl. Abb. 6.1 (S. 122), lassen sich als *Sammon-Abbildung* der Kodebuchvektoren einer 12×10 -SOM veranschaulichen. In Abb. 4.2 ist sie mit der SOM-Gitterstruktur und den abgekürzten Namen der Arten gemäß Tab. 6.1 (S. 121) dargestellt.²⁾

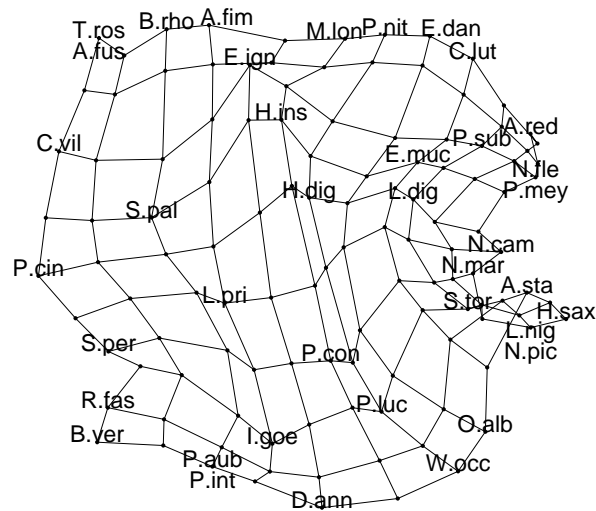


Abbildung 4.2: Sammon-Abbildung von Kodebuchvektoren einer SOM mit 12×10 Neuronen, basierend auf jährlichen Abundanzmustern von 40 EPT-Arten über 30 Jahre in der Emergenzfalle „Haus B“ am Breitenbach; die Abkürzungen sind in Tab. 6.1 (S. 121) erklärt.

In dieser Darstellung sind keine Ausreißer unter den Kodebuchvektoren zu erkennen. Ihre Abstände zu den im Training verwendeten Daten waren nie größer als die Summe aus dem Mittelwert des Quantisierungsfehlers und dem Dreifachen seiner Standardabweichung.³⁾ Daher ist nicht davon auszugehen, dass ein Abundanzmuster durch keinen der Kodebuchvektoren angenähert wurde und völlig von den anderen abwich.

Durch die Anwendung von GRNNs und einem schrittweisen Verfahren sowie Genetischen Algorithmen wurden für die Prognosen der jährlichen Gesamtzahl von Individuen der Köcherfliege *Tinodes rostocki* in der Falle „Haus B“ einige Prädiktoren als besonders relevant eingestuft,⁴⁾ darunter die Abundanz ihrer Elterngeneration, die Regenmengen im

²⁾ Eine ökologische Interpretation dieses Ergebnisses findet sich in [OWWS01, S. 210 f.].

³⁾ Diese Aussage ergibt sich aus Berechnungen der Quantisierungsfehler, die in der Datei `l_som_output.smf` als Ausgabe des Programms `visual` gespeichert sind. Der mittlere Quantisierungsfehler lag bei 0.53, die Standardabweichung betrug 0.20.

⁴⁾ Der Genetische Algorithmus wurde im Anschluss an das schrittweise Verfahren angewendet, um die Auswahl von Prädiktoren zu verbessern. Allerdings konnte dieses Ziel auch nach 8400 Generationen mit 10 Individuen pro Generation, einer Mutationsrate von 0.08, einer Crossover-Rate von 0.8 und einem Strafterm (Penalty) für jeden Prädiktor von 0.015 nicht erreicht werden [OWWS01, S. 212].

Juli und April, der Niederschlag im Dezember, die maximale Lufttemperatur im Januar sowie der maximale Abfluss des Breitenbaches im Juni des betreffenden Jahres.

Ein RBFSOM-Netz mit einer Kette aus 10 SOM-Neuronen bzw. RBF-Zentren wurde mit diesen Eingaben trainiert. Die Darstellungen in Abb. 4.3, bei denen jeweils zwei der sechs Prädiktoren in dem Wertebereich der Lerndaten variiert und die restlichen konstant auf ihrem Mittelwert gehalten wurden, zeigen in diesem Fall nahezu lineare Funktionen. Unter der Voraussetzung der Gültigkeit des Modells verdeutlichen sie, dass der Niederschlag im Dezember unter sonst als normal oder mittelmäßig zu bezeichnenden Bedingungen einen weitaus geringeren Einfluss auf die Häufigkeit dieser Köcherfliegenart am Breitenbach hat als der Niederschlag im April. Hingegen bewirken die Abundanz der Elterngeneration und die Regenmenge im Juli gleichermaßen starke Veränderungen in der Populationsdichte. Diese nimmt in Abhängigkeit von den beiden genannten Prädiktoren die höchsten Werte an, wenn eine große Elternzahl und ein regenarmer Juli auftreten. In Jahren, in denen eine verhältnismäßig schwach auftretende vorige Generation und ein regenreicher Juli zusammentreffen, sind nur wenige Exemplare von *Tinodes rostocki* zu erwarten.

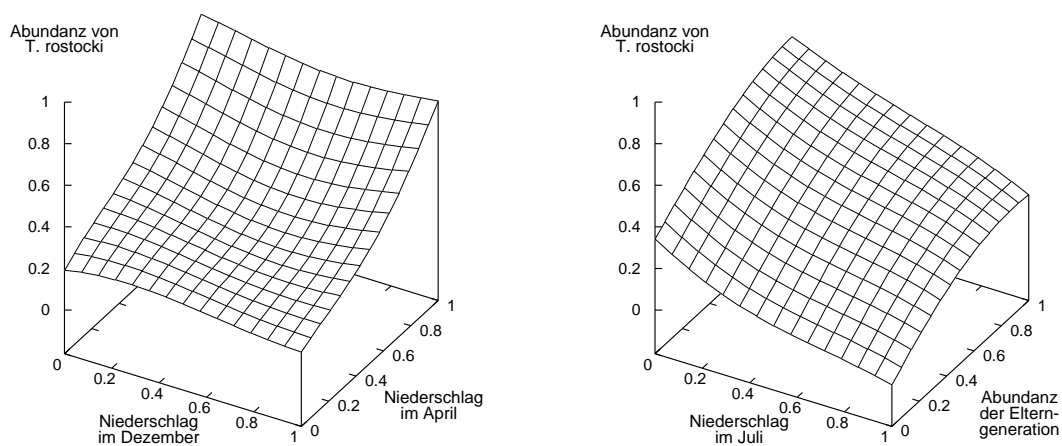


Abbildung 4.3: Visualisierung der Abhängigkeit der jährlichen Abundanz von *Tinodes rostocki* (1969–1998) von je zwei Variablen auf der Basis eines RBFSOM-Netzwerk-Modells (die Trainingsdaten wurden jeweils auf $[0,1]$ abgebildet).

Die Prognosen der Testdaten durch das RBFSOM-Netz lieferten einen RMSE von 0.1172, vgl. Abb. 4.4 (links). Die beste durch ein LNN gegebene lineare Approximation, die nach einer schrittweisen Prädiktorenauswahl auf nur vier unabhängigen Größen basierte, erzeugte auf den Testdaten einen RMSE von 0.1363, vgl. [OWWS01, S. 211].

Bei der Beurteilung der Güte eines Modells lassen sich die Generalisierungsfehler, hier der RMSE, an anderen Größen relativieren. Zum Beispiel war die Abweichung zwischen der Prognose und den beobachteten Werten für die jährliche Gesamthäufigkeit der Steinfliege *Amphinemura standfussi* für sechs Jahre im Test — wie Abb. 4.4 (rechts) verdeut-

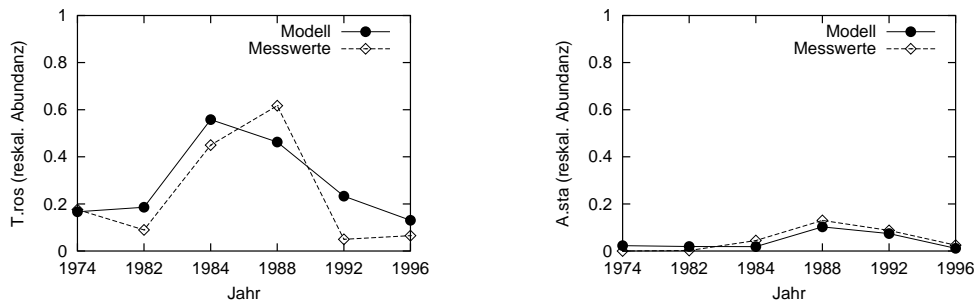


Abbildung 4.4: Vorhersage der $[0,1]$ -skalierten jährlichen Abundanzen von *Tino-des rostocki* (links) und *Amphinemura standfussi* (rechts) für sechs Jahre im Test. Die Verbindungslinien zwischen den Datenpunkten dienen bei dieser und den anderen Grafiken in diesem Abschnitt ausschließlich der besseren Erkennbarkeit.

licht — mit einem RMSE von 0.0207 sehr gering. Setzt man mit

$$E_{\text{rel}}^{\bar{y}} = \frac{\sqrt{\sum_{i=1}^n (y_i - \hat{y}_i)^2}}{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}, \quad (4.1)$$

vgl. [Mas93, S. 65] und für $E_{\text{rel}}^{\bar{y}^2}$ [Bis95, S. 197], den RMSE mit der Standardabweichung ins Verhältnis, ist der Unterschied zwischen den Werten von 0.44 für *A. standfussi* und 0.54 für *T. rostocki* geringer. Da beide Werte deutlich kleiner als eins sind, erweisen sich die Modelle — zumindest für die wenigen Testdaten — als der Mindestanforderung \bar{y} überlegen. Die Bestimmtheitsmaße ergeben sich mit $B = 1 - E_{\text{rel}}^{\bar{y}^2}$ zu 0.81 für die Stein- und 0.71 für die Köcherfliege.

Zur Vorhersage der jährlichen Anzahlen von Individuen der Köcherfliege *Apatania fimbriata* in der Emergenzfall „Haus B“ (siehe Abb. 4.1) wurden allein die Muster des maximalen monatlichen Abflusses der 13 Monate von Juni des Vorjahres bis zum Juni des aktuellen, der Bezeichnung dienenden Jahres verwendet, die den Zeitraum für einen Lebenszyklus der Köcherfliege überdeckten. Wegen des geringen Datenumfanges kam eine als Kette angeordnete SOM zum Einsatz, die nur sechs Neuronen enthielt. Die durch die Kodebuchvektoren gegebenen Prototypen für die Klassen von Mustern sind in Abb. 4.5 den zur Überprüfung der Generalisierungsfähigkeit verwendeten Test-Eingabevektoren gegenüber gestellt.⁵⁾

Die aufgrund des kleinsten euklidischen Abstandes gegebene Zuordnung der Testmuster zu den Prototypen, vgl. Tab. 4.1, ergab z_2 als „Sieger“ für die betreffenden 13 Monate in 1973/74 und 1991/92. Das Abflussmuster für 1981/82 war am ähnlichsten zu z_5 , dasjenige von 1983/84 fiel in die durch z_3 bestimmte Klasse, das von 1987/88 ließ sich durch

⁵⁾ Die Übereinstimmung der Anzahlen der SOM-Neuronen und der Testdatensätze ist hier ohne Bedeutung.

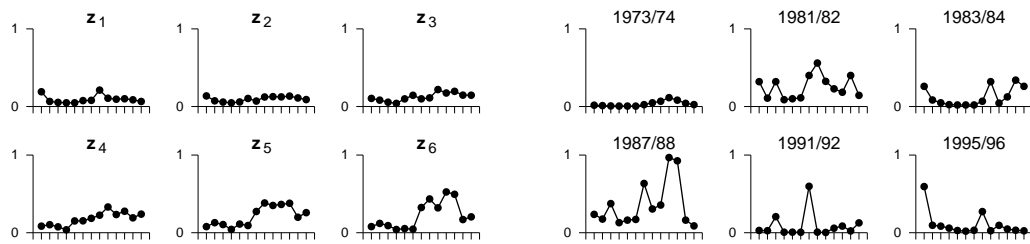


Abbildung 4.5: Prototypen (Kodebuch-Vektoren) für Klassen von (reskalierten) Abflussmustern während des Lebenszyklus von *Apatania fimbriata* (links) und Test-Eingabemuster (rechts). Die Abszissen beschreiben jeweils einen Zeitraum von 13 Monaten vom Juni des Vorjahres bis zum Juni des Jahres, für das die Abundanz der Imagines der Köcherfliege prognostiziert wurde. Beispielsweise bedeutet 1973/74 den Zeitraum von jeweils einschließlich Juni 1973 bis Juni 1974.

z_6 repräsentieren und die 1995/96 beobachteten monatlichen Abflussmaxima konnten am besten durch z_1 angenähert werden. Keines der im Testdatensatz vorhandenen Abflussmuster glich z_4 .

Bezüglich der sechs Testdatensätze differierten die Messwerte der jährlichen Abundanzen von *A. fimbriata* und die Prognosewerte bei der Verwendung der SOM-Kodebuchvektoren als RBF-Zentren in dem RBFSOM-Netzwerk auf der Basis der Abflussmuster am stärksten für die Jahre 1984 und 1988, wie aus den in Abb. 4.6 dargestellten *Box-Whisker-Plots* [KSV92, S. 112] ersichtlich ist.

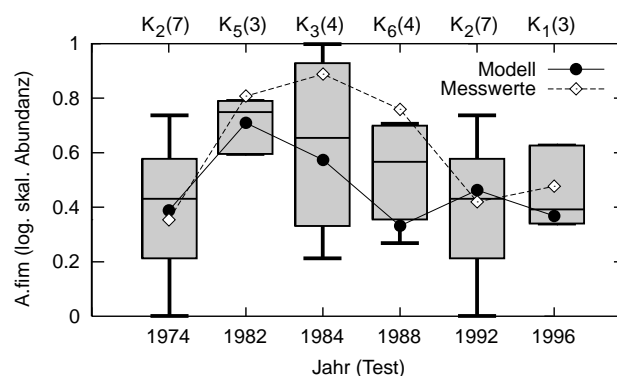


Abbildung 4.6: Vorhersage der logarithmisch reskalierten jährlichen Abundanz von *Apatania fimbriata* mit Vergleich zwischen Messung und Prognose sowie die durch *Box-Whisker-Plots* dargestellten Verteilungen der Ausgabegröße in den Klassen \mathcal{Y}_j^L . Oben sind die durch sechs SOM-Neuronen (mit Kodebuchvektoren z_1, \dots, z_6 , vgl. Abb. 4.5) gebildeten Klassen \mathcal{X}_j^L , in die jede Test-Eingabe fällt, als K_j angegeben. Dahinter stehen in Klammern die Anzahlen der in den betreffenden Klassen liegenden Trainingseingabedaten.

Bei gleicher Anzahl von Trainingseingabemustern in den Einzugsgebieten der jeweils zugehörigen SOM-Neuronen war die Streuung der Ausgabegröße für 1984 wesentlich größer als für 1988. Der außerhalb des Bereiches der Trainingsdaten liegende Testdatenpunkt für 1988 kann durchaus ein tendenziell außergewöhnlicher Wert sein und in diesem Fall ist der Klassenmittelwert oder der Median für die Klasse eine bessere Schätzung. Für Prognosen, deren Eingaben in die dritte Klasse fallen, sind aufgrund der großen Streuung keine guten Ergebnisse zu erwarten. Die Netzausgaben für die Jahre 1974 und 1992 liegen sehr nahe am Messwert und zudem nahe am Klassenmittel der Trainingsdaten. Für beide Testdatensätze gehören die Eingabemuster zur umfangreichsten Klasse, in der die Werte der Ausgabegrößen allerdings stark streuen. Der erste Testdatensatz (1974) liefert den kleinsten Fehler (siehe Abb. 4.6), gehört gemäß Tab. 4.1 der zweiten Abundanzmuster-Klasse mit der größten Anzahl von Stützpunkten an, erzeugt im Vergleich zu den anderen Test-Eingabemustern den kleinsten Abstand zum nächsten Kodebuchvektor im SOM-Netz und die größte Aktivität. Die zugehörigen jährlichen Abundanzen von *A. fimbriata* streuen in den zum Lernen verwendeten Daten aber beträchtlich. Ähnliches gilt für das fünfte Testmuster 1992.

Tab. 4.1 zeigt die Netzausgaben und einige Qualitätsmaße des RBFSOM-Modells für die Testdaten. Die Prognosen sind für fast alle Testdaten durch das Klassenmittel ein

Tabelle 4.1: Ausgaben von *Visualrbfn* in der Datei *work_results.txt*, vgl. Tab. 3.3 (S. 58). In den Spalten stehen: Nummer des Testdatensatzes i , Jahr, Netzausgabe \hat{y}_i (A), Messwert y_i (B), Residuum $y_i - \hat{y}_i$ (C), maximale Erregung aller RBF-Neuronen für die Netzeingabe \mathbf{x}_i (D), $\kappa(\mathbf{x}_i)$ (E), Quantisierungsfehler (F), $\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$ (G), Spannweite $\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L) - \min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ (H), $\min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ (I), $\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ (J), Mittelwert der Elemente von $\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L$ (K).

i	Jahr	A	B	C	D	E	F	G	H	I	J	K
1	1974	0.38	0.35	-0.03	0.88	2	0.17	7	0.73	0	0.73	0.40
2	1982	0.70	0.80	0.10	0.84	5	0.37	3	0.20	0.59	0.79	0.71
3	1984	0.57	0.88	0.31	0.83	3	0.26	4	0.78	0.21	1	0.63
4	1988	0.33	0.75	0.42	0.74	6	0.57	4	0.43	0.27	0.71	0.52
5	1992	0.46	0.42	-0.04	0.69	2	0.43	7	0.73	0	0.73	0.40
6	1996	0.36	0.47	0.11	0.74	1	0.31	3	0.28	0.34	0.63	0.45

wenig besser als durch das RBFSOM-Modell. Auch die Mediane der Klassen liegen bis auf das Testmuster 1974 zwischen den Testdaten und den RBFSOM-Ausgaben, was aus Abb. 4.6 ersichtlich ist. Das Testdatenmuster mit der Bezeichnung 1982 (Testvektor 2) weist nicht nur eine geringe Abweichung von Beobachtung und Prognose auf, sondern seine Eingabe gehört zu der Klasse 5 von Mustern, die sich durch eine geringe Streuung der Ausgabegröße auszeichnet, die allerdings nur durch drei Trainingsdatensätze gestützt

wird. Das Muster liegt nicht sehr weit vom CBV entfernt (siehe Spalte F), hat die kleinste Spannweite im Testdatensatz, vgl. Tab. 4.1 und Abb. 4.6, und liegt im Interquartilbereich, d. h. zwischen dem unteren und dem oberen Quartil [KSV92, S. 34] bzw. dem 25%- und 75%-Perzentil [Bor93, S. 40 f.].

Vergleicht man die großen Streuungen für die Klassen 2 und 3 in Abb. 4.6 und zieht man die augenscheinliche Ähnlichkeit der ersten drei Klassen von Abflussmustern in Betracht, resultiert daraus die Feststellung, dass man für Jahre mit dauerndem Niedrigabfluss keine zuverlässigen Prognosen über die Abundanz von *A. fimbriata* abgeben kann. Hingegen könnten Jahre mit einem zu z_5 ähnlichen Abflussmuster mit deutlichem, aber nicht sehr starkem Abfluss im Winterhalbjahr verhältnismäßig gute Abundanzprognosen zulassen. Verschiedene Möglichkeiten der Visualisierung der Daten und des Netzwerkverhaltens durch das RBFSOM-Modell als Ausgaben der Software *Visualrbfn* zeigt Abb. 4.7. Dabei

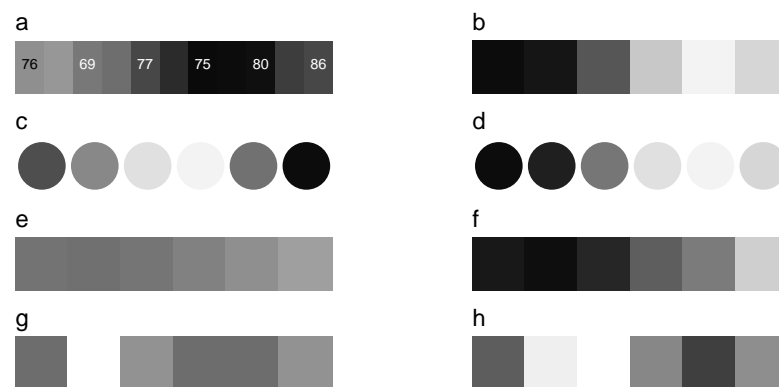


Abbildung 4.7: Visualisierung der SOM-Neuronen mit Kodebuchvektoren z_1 bis z_6 (von links nach rechts) in der verdeckten Schicht eines RBFSOM-Netzwerkes; Beispiel von Daten mit *Apatania fimbriata*; U-Matrix-Darstellung, beschriftet mit den letzten beiden Ziffern der kleinsten Jahreszahl der Jahre, deren Muster monatlicher (vom Juni des Vorjahres bis zum Juni des betreffenden Jahres betrachtete) Abflussmaxima des Breitenbaches an „Haus B“ zu der jeweiligen Klasse gehören (a); Netzantwort des RBFSOM-Netzes für eingetragene Prototypen (b); Komponentenebenen für den maximalen Abfluss im November (c) und im Februar (d); Abstände der Kodebuchvektoren zum Test-Eingabemuster 1991/92 (e); Aktivitäten der RBF-Neuronen über der SOM für den Testzeitraum in 1987/88 (f); Anzahl von Lerndaten für jede Klasse, so reskaliert, dass 0 durch Schwarz und 7 durch Weiß repräsentiert würde (g); Spannweiten der Abundanz von *A. fimbriata* für jede Klasse (h).

sind dunkle Graustufen niedrigen Werten zugeordnet und helle Graustufen deuten auf hohe Werte hin. Die U-Matrix-Darstellung (siehe S. 16) veranschaulicht die Abstände zwischen Kodebuchvektoren benachbarter SOM-Neuronen auf der Basis verschiedener Graustufen (a). Die Beschriftung 76 bedeutet beispielsweise, dass 1976 — bzw. das Muster des maximalen monatlichen Abflusses des Breitenbaches von jeweils einschließlich Juni 1975 bis Juni 1976 — das erste Jahr in den Trainingsdaten war, das in die Klasse mit

dem ersten Kodebuchvektor ähnlichen Abflussmustern fiel. Das Muster des ersten Jahres (1969), das zum Lernen des Netzwerkes verwendet wurde, glich am stärksten dem durch z_2 repräsentierten Typ von Abflussmustern.

Je heller der Grauton zwischen den SOM-Elementen ist, desto größer ist die Ähnlichkeit (da der euklidische Abstand kleiner ist) zwischen den zugehörigen Kodebuchvektoren. Beispielsweise sind die Muster z_1 und z_2 (mit 1969 und 1976 zu vergleichen) ähnlicher als z_5 und z_6 (z. B. 1980 bzw. 1986). Abb. 4.7 (b) zeigt die erwarteten Netzausgaben für die jeweiligen Kodebuchvektoren als Netzeingaben. Maximale Abundanzen von *A. fimbriata* werden für Abflussmuster vorhergesagt, die dem Prototypen z_5 ähneln (vgl. Abb. 4.5, links). Für Jahre mit durchweg niedrigem Abfluss, wie sie ebenfalls in den oberen drei Mustern der linken Grafik in Abb. 4.5 zu sehen sind, prognostiziert das Netzwerk wenige Imagines der Köcherfliegenart. Die Komponenten-Ebenen (c) und (d) stellen eine Möglichkeit dar, um die Werte der Komponenten der Kodebuchvektoren mit Hilfe von Graustufen auf dem SOM-Gitter zu illustrieren. Das Beispiel (c) verdeutlicht, dass der Abfluss im November (6. Komponente, siehe Abb. 4.5) für das vierte Muster (ähnlich zu 1975) im Trainingsdatensatz maximal ist. Die stärksten monatlichen Abflüsse im Februar (Vektorkomponente 9) waren in der Klasse 5, die u. a. den Trainingsdatensatz für 1980 enthält, höher als bei den Prototypen aller anderen Klassen (d). Abb. 4.7 (e) zeigt die Abstände des Testmusters für 1992 zu den Kodebuchvektoren. Aufgrund des hellen Grautons wird ersichtlich, dass die größte euklidische Distanz zu den Elementen der Klasse 6 besteht. Die Zuordnung zu der Klasse 2 ist aufgrund der großen Ähnlichkeiten der Elemente der ersten drei Klassen schwachen Grauton-Abstufungen sehr undeutlich (vgl. Abb. 4.5). Die Aktivitäten der RBF-Neuronen hängen von den Abständen der Kodebuchvektoren (RBF-Zentren) und den Testmustern ab (f). Die maximale Aktivität für das Jahr 1988 zeigt sich beispielsweise deutlich für das letzte Neuron der Kette und dem Prototypen z_6 . Die Anzahlen von Trainingsdaten, die dem jeweiligen SOM-Neuron zugeordnet sind, sind in Abb. 4.6 als Zahlen (in Klammern) sowie in Abb. 4.7 (g) durch Graustufen wiedergegeben, siehe auch Spalte G in Tab. 4.1. Die Spannweiten der in der Emergenz-fälle gefundenen Abundanzen von *A. fimbriata*, die der Länge der „Whiskers“ in Abb. 4.6 entsprechen, finden sich für jede Klasse in Grafik (h). Hier ist festzustellen, dass dieses Streuungsmaß vor allem für die Klasse 5 einen verhältnismäßig geringen Wert annimmt.

4.1.4 Diskussion

Ein bedeutender Teil der ökologischen Wissenschaft beschäftigt sich damit zu verstehen, was die Häufigkeit der Organismen bestimmt [BHT91, S. 563]. Modelle zur Vorhersage der Abundanzen von Organismen in Abhängigkeit von beobachteten Werten wesentlicher Umweltfaktoren sowie die nur durch Freilanduntersuchungen zu gewinnende Kenntnis über die realisierte Nische einer Art und die Beschreibung der Lebensgemeinschaften liefern nicht nur grundlegende ökologische Erkenntnisse, sondern können überdies einen Beitrag zum Arten- und Umweltschutz leisten.

In der Praxis sind die hochdimensionalen Hypervolumen,⁶⁾ die eine Nische definieren, jedoch bestenfalls als Projektionen auf wenige Achsen zu beschreiben. Viele der wirksamen Umweltfaktoren und Ressourcen sind kaum bekannt oder entziehen sich einer Messung. In der vorliegenden Fallstudie wurden Ansätze zur Beantwortung der grundlegenden Frage, warum eine Art an einer Stelle mit einer bestimmten Populationsdichte vorkommt und welche Faktoren Fluktuationen in den Häufigkeiten der Arten verursachen [BHT91, S. 563] durch die Beschreibung stochastischer Abhängigkeiten gegeben. Dabei ist die Zusammensetzung der Lebensgemeinschaft^(e) des Ökosystems Breitenbach und die Abschätzung der Häufigkeiten der Arten mit diesen Daten aber nur teilweise beschreibbar, da die Interaktionen innerhalb der Biozönose in Form von inter- und intraspezifischer Konkurrenz und Räuber-Beute-Beziehungen sowie Zu- und Abwanderung fehlen.⁷⁾

Der Abfluss kann als wesentlicher, sowohl über die Strömung und den Wasserstand unmittelbar als auch indirekt wirkender Umweltfaktor für die Biozönose des Breitenbaches angesehen werden. Er verändert durch die Umgestaltung des Bachbettes den Lebensraum (Steine, Kies, Schlamm), die Nahrung (u. a. Biofilm, Detritus), die chemischen Eigenschaften (Säurebindungsvermögen, pH-Wert) und nicht zuletzt ist er für die Drift der Individuen verantwortlich. Ob die Dauer, die Häufigkeit oder andere Merkmale von Hochwasserereignissen den größten Einfluss auf die Lebensgemeinschaft ausüben, bleibt zu überprüfen.

Für die Köcherfliege *Tinodes rostocki* wurden statistisch relevante Umweltfaktoren gefunden, um die Abundanz der Art an der untersuchten Probestelle mit guter Qualität vorherzusagen. Über die Wirkungsweise der Größen auf den als Abundanz gemessenen Erfolg der Art lassen sich Hypothesen aufstellen.⁸⁾ Ein niedriger Abfluss im September, dem Monat mit den ohnehin im langfristigen Mittel niedrigsten Wasserständen, verringert das zur Verfügung stehende Habitat der Larven, während ein mäßiger Abfluss im April das Vertrocknen der sich an der Wasserstandslinie befindenden, unbeweglichen Puppen verhindert.

Die Häufigkeit der Köcherfliege *Apatania fimbriata* wurde nur auf der Basis maximaler monatlicher Abflusswerte prognostiziert. Mittelwerte und Minima sowie andere möglicherweise wichtige Einflussgrößen wurden vernachlässigt. Es ist anzunehmen, dass die Variabilität der Daten in den einzelnen Klassen durch die Wahl weiterer Prädiktoren verringert werden kann.

Die Sammon-Abbildung der Prototypen jährlicher Abundanzen von EPT-Arten (vgl. Abb. 4.2 auf S. 70) erweitert die Korrelationsanalysen in [Oba98, S. 38 ff.], frühere Anwendun-

⁶⁾ Während die Beschreibung einer Nische bezüglich der möglichen Präsenz oder Absenz einer Art in Abhängigkeit der Umweltfaktoren und Ressourcen als Klassifikationsaufgabe aufgefasst werden kann, für die sich neben KNNs die Ansätze der Fuzzy-Mengen anbieten, wurde hier versucht, die Abundanz von Arten als Maß für die Populationsdichte als Regression zu schätzen.

⁷⁾ Auch die Umkehrung der Aufgabenstellung, das als *Bioindikation* bezeichnete Schließen von einer beobachteten Biozönose auf die momentan oder vorher herrschenden Lebensbedingungen, ist mit großen Schwierigkeiten verbunden.

⁸⁾ Neben der Häufigkeit ist auch die Biomasse ein denkbare Erfolgsmaß einer Population.

gen von SOMs auf diesem Gebiet [Foo99, CPMC96] sowie die Methoden zur *Ordinierung* von Lebensgemeinschaften, wobei Gesellschaften so in einem Diagramm dargestellt werden, dass ähnliche nahe zusammenliegen [BHT91, S. 689].⁹⁾ Liegen die Abundanzmuster von mehreren EPT-Arten gemäß der Sammon-Abbildung der Kodebuchvektoren im rezeptiven Feld eines einzigen Neurons, besteht eine verhältnismäßig große Ähnlichkeit. Dann kann die Bildung mehrerer Prognosemodelle unnötig sein, da die relative Häufigkeit einer Art durch die einer anderen mit einem ähnlichen Abundanzmuster beschrieben wird. Trotzdem sind vergleichbare Populationsschwankungen noch kein Nachweis der Existenz interspezifischer Abhängigkeiten, sondern möglicherweise liegen ähnliche Umweltansprüche der Taxa vor. Zur Veranschaulichung der Ähnlichkeiten der Abundanzmuster und zur Ausreißer-Erkennung können die Daten anstatt über die Zuordnung zu SOM-Kodebuchvektoren auch direkt durch die Sammon-Abbildung projiziert werden.

Bei Freilandbeobachtungen und Langzeitstudien kann das *Ceteris-paribus-Prinzip*, nach dem man versucht, alle im Versuch unberücksichtigten Faktoren, die Einfluss auf die Messergebnisse haben können, möglichst konstant zu halten [KSV92, S. 215], prinzipiell kaum berücksichtigt werden. Im Untersuchungszeitraum der vorliegenden Fallstudie änderten sich einige Umweltbedingungen. Beispielsweise wurden weite Teile des Breitenbach-Einzugsgebietes unter Naturschutz gestellt, die Beweidung der angrenzenden Wiesen durch Kühe wurde eingestellt, die forstliche Nutzung veränderte sich, gelegentlich wurde der Bach zumindest stellenweise von in ihm und über ihn wuchernder Vegetation befreit und die Messverfahren wurden dem technischen Stand der Zeit angepasst. Die durch die zufällige Auswahl für das Netztraining verwendeten Daten des Jahres, in dem sich am Breitenbach ein schwerwiegender Unfall mit Insektengift ereignete [Zwi93], der die Populationen vieler Arten kurzzeitig in Gefahr brachte, sind nicht als Ausreißer zu erkennen.

Es ist anzunehmen, dass aufgrund der großen Zahl von Eiern pro Weibchen — die Eizahl kann bei Köcherfliegen einige Dutzend bis mehrere Hundert [Zah89, S. 61], bei Eintags- und Steinfliegen je nach Art sogar bis zu mehreren Tausend [Grz93, S. 75, 79] betragen — ein großes Fortpflanzungspotential besteht, sodass sehr wenige Individuen einer EPT-Art ausreichen, um in der folgenden Generation den Lebensraum zu „sättigen“. Weiterhin können fliegende Insekten Lebensräume sehr schnell (wieder-)besiedeln und Larven können von anderen Bachabschnitten verdriftet werden.

Eine gute Anpassung gegenüber *Störungen* kann einen Selektionsvorteil bieten. Welche Klasse von Abflussmustern, deren Prototypen in Abb. 4.5 dargestellt sind, Störungen enthält, ist nicht eindeutig zu beantworten, denn die im Sinne von extremen Abflussereignissen ohne Störungen verlaufenden Abflussmuster in der oberen Zeile der Zusammenstellung von Grafiken, kann eine Störung im Sinne des *Disturbance-Konzeptes* [Pof92] bedeuten. Beispielsweise kann das Abflussmuster im Jahr 1992 bezüglich der Neuartigkeit stärker von den für die betreffende Jahreszeit zu erwartenden Bedingungen abwei-

⁹⁾ Die Ergebnisse einer Analyse mit Ordinierungsverfahren von 34 Invertebratengesellschaften in Flüssen von Südeuropa sowie von 68 benthischen Invertebratengesellschaften im norwegischen Oslofjord finden sich in [BHT91, S. 692 ff.].

chen als beispielsweise das stark variierende Muster von 1988. Daher kann auch ein ausbleibendes Ansteigen des Abflusses für einzelne Arten störend wirken, je nachdem, in welchem Stadium des Lebenszyklus die Individuen einer Art durch ungewöhnliche Ereignisse betroffen werden. Der im Vergleich zu den anderen Testdaten größte Abstand zu dem nächstliegenden SOM-Kodebuchvektor in Tab. 4.1 deutet darauf hin, dass 1988 das ungewöhnlichste Abflussjahr im Untersuchungszeitraum war.

Das 1971 publizierte Konzept des *Emergenzfanges* [III71] von Insekten an Fließgewässern erleichtert die Bestimmung der aus dem Bach kommenden, in der Regel erwachsenen Individuen, vermindert die Veränderung des Bachbodens und damit des Habitats für den wesentlichen Teil der Lebensgemeinschaft und sichert, dass die Tiere unabhängig von den lokalen Substraten oder Tiefen des Baches aus einem definierten Teil der Oberfläche des Gewässers stammen [Mal02]. Versucht man, die gelegentlich angezweifelte Repräsentativität [Mal02] der Emergenzmethode für die Abundanzschätzung der verschiedenen Taxa generell und insbesondere für die verwendete Falle zu testen, müsste eine andere repräsentative Methode zur Verfügung stehen, was aber nicht gegeben ist. Für die vorliegende Untersuchung wird vorausgesetzt, dass der Zuflug (Import) in den Breitenbach und speziell in die Emergenzfallen gering ist und die Fangmethode die Populationsgröße mit ausreichender Güte schätzt.

Ein grundsätzliches Problem für die Modellbildung in dieser Fallstudie stellt der geringe Datenumfang dar. Obwohl die Daten unter einem enormen Aufwand (u. a. nahezu tägliches Absammeln der Fallen, Bestimmung des Fanges, Messung vieler abiotischer Größen über mehr als drei Jahrzehnte) erhoben wurden, ergeben sich für *univoltine* Arten, die nur eine Generation pro Jahr hervorbringen, nur 30 Datensätze, die überdies in Trainings-, Validierungs- und Testdaten eingeteilt werden müssen. Daher liegen nur wenige Trainingsdaten vor, um die Parameter des Netzes anzupassen und die wichtigsten Eingabe-Größen zu bestimmen. Mit einer geringen Anzahl von Testdaten können überdies keine zuverlässigen Abschätzungen der Güte des Modells gefunden werden. Bei der Prognose der Abundanzen von *A. fimbriata* auf der Basis der Abflussmuster des Breitenbaches über 13 Monate erlauben die wenigen Daten kaum eine größere Anzahl von Klassen als sie in der Fallstudie gewählt wurde, da sonst triviale Zuordnungen zu erwarten sind, bei denen nur jeweils ein Datensatz in eine Klasse fällt oder SOM-Neuronen für keinen Eingabevektor Sieger sind. Bei Abb. 4.6 ist fraglich, wie zuverlässig die Aussagen bei wenigen Punkten insgesamt und pro Klasse sein können.

Bei der Verwendung vieler Netzwerke und zahlreicher Arten kann es vor allem bei der Verwendung weniger Datensätze zum Testen der Generalisierungsfähigkeit zum Lernen zweiter Art [Klö94] kommen, bei dem ein zufällig geringer Fehler bei den Testdaten ein gutes Modell zur Beschreibung des Zusammenhanges vortäuscht. Einen Ausweg können Kreuzvalidierungen bieten.

Weitere Aspekte zu dieser Fallstudie werden in der Abschlussdiskussion in Kapitel 5 ab S. 111 erläutert.

4.2 Wetter-Abfluss-Beziehung für einen Bach und sein Einzugsgebiet

4.2.1 Einführung

Die *Wasserstände von Fließgewässern* waren seit jeher für die an und von ihnen lebenden Menschen von großer Bedeutung, wovon schon die ältesten dokumentierten Zeitreihen der jährlichen niedrigsten Wasserstände des Nils von 622–1470 n. Chr. zeugen, vgl. [And84, S. 5].¹⁰⁾ In den letzten Jahrzehnten kosteten Überschwemmungskatastrophen in einigen Ländern viele Menschenleben und verursachten zudem große Sachschäden. Ein Beispiel aus jüngster Zeit ist das „Jahrhunderthochwasser“ der Elbe und ihrer Zuflüsse im August 2002. Zuverlässige Prognosen sind nicht nur für die von Überflutungen und Tiefständen betroffene Bevölkerung wichtig, sondern sie sind auch für Limnologen und Hydrologen von Interesse. Der eng mit dem Wasserstand zusammenhängende *Abfluss*^(e), der auch als *Schüttung* bezeichnet wird, ist vor allem in und an chemisch und strukturell unbelasteten Fließgewässern ein wesentlicher Umweltfaktor für die bakteriologischen, pflanzlichen und tierischen Lebensgemeinschaften, siehe u. a. [RHST94].

Konzeptionelle Modelle zur Abflussvorhersage versuchen, die Teilprozesse und Mechanismen in Fließgewässern und ihren Einzugsgebieten deterministisch abzubilden. Die Schwierigkeiten bei ihrer Erstellung liegen u. a. in der Implementierung und Kalibrierung, den benötigten komplizierten mathematischen Verfahren, der großen Menge von Daten und dem nötigen Expertenwissen [ZBS99, S. 33]. Prognosen aufgrund von *Zeitreihenanalysen* sind in der Regel einfacher zu gewinnen, da die zugrunde liegenden Prozesse nicht explizit bekannt sein und modelliert werden müssen [ZBS99, S. 33]. Einige Anwendungen von Modellen für ARMA-Prozesse in der Gewässerparameter-Vorhersage sind in [MD96] zu finden.

Die Eignung von KNNs zur Prognose des Abflusses und des Wasserstandes für Fließgewässer ist aus zahlreichen Publikationen belegt, u. a. [ZFH94, LS95, LDDEG96, FJ98, JF98]; Überblicke sind in [MD96, ZBS99, MD00] enthalten.

In der vorliegenden Fallstudie steht die Anwendung des RBFSOM-Netzwerkes durch die Software *Visualrbfn* im Vordergrund. Dabei wurden zur Prognose des mittleren täglichen Abflusses eines kleinen Baches zwei *Wetter-Abfluss-Modelle*¹¹⁾ erstellt. Zunächst dienten nur Informationen über die Temperatur- und Niederschlagsverhältnisse im Einzugsgebiet des Fließgewässers über verschiedene Zeiträume als Prädiktoren, während anschließend auch vorangegangene Abflusswerte als bekannt vorausgesetzt wurden.

¹⁰⁾ Für den Zeitraum 715–1284 n. Chr. sind die jährlichen niedrigsten Wasserstände des Nils in [SS97, S. 532] zu finden.

¹¹⁾ In der Fachliteratur und im Internet findet man Informationen über solche Modelle, vor allem den Spezialfall der Niederschlags-Abfluss-Modelle, bei der Suche nach *rainfall and stream flow model*, *precipitation-runoff model*, *precipitation-discharge model* und Kombinationen dieser Begriffe.

4.2.2 Datengrundlage und Vorverarbeitung

Die vorliegende Fallstudie basiert auf Daten der täglichen Niederschlagsmengen an den Wettermess-Stationen 51033 bei Michelsrombach und in der Stadt Schlitz sowie den am Breitenbach (siehe S. 68) an der Probestelle „Haus B“ gemessenen Tagesmittelwerten der Lufttemperatur (im Schatten) und des Abflusses.

Hochwasserereignissen mit einer teilweise nur grob geschätzten Schüttung von mehr als 120 ℓ/s im Tagesmittel, bei der das Gewässer über die Ufer trat, wurde dieser Wert zugewiesen, um den starken, oft verfälschenden Einfluss einzelner Außenpunkte auf die Regressionsschätzung abzuschwächen.

Die Messwerte dieser Größen können in der genannten Reihenfolge als Komponenten einer *multivariaten Zeitreihe* $\mathbf{u}(t) = (u_1, u_2, u_3, u_4)(t)$ aufgefasst werden. Der *Zeitparameter* t dient der Nummerierung der einzelnen Tage.

Im Rahmen einer Vorverarbeitung der Daten wurden mit

$$\mathbf{u}^{t_1-t_2}(t) := \sum_{\tau=t-t_1}^{t-t_2-1} \mathbf{u}(\tau) \quad (4.2)$$

die Teilsummen $\mathbf{u}^{0-1}(t)$, $\mathbf{u}^{1-2}(t)$, $\mathbf{u}^{2-3}(t)$, $\mathbf{u}^{3-4}(t)$, $\mathbf{u}^{4-5}(t)$, $\mathbf{u}^{0-7}(t)$, $\mathbf{u}^{7-14}(t)$, $\mathbf{u}^{0-14}(t)$, $\mathbf{u}^{14-28}(t)$, $\mathbf{u}^{0-28}(t)$, $\mathbf{u}^{28-56}(t)$, $\mathbf{u}^{0-80}(t)$ und $\mathbf{u}^{80-160}(t)$ gebildet.¹²⁾ Alle Größen wurden anschließend [0,1]-skaliert.

Da die gebildeten Summen nach dieser Skalentransformation nicht mehr von entsprechend skalierten Mittelwerten zu unterscheiden sind, ist es zulässig, im Folgenden gelegentlich auch beispielsweise von der mittleren Temperatur in den beiden Wochen vor dem Prognosezeitpunkt t anstatt von der Temperatursumme über den genannten Zeitraum zu sprechen.

Die Tagesmittel der Lufttemperatur schwankten im Untersuchungszeitraum zwischen dem Tiefstwert -15.8 °C und dem Höchstwert $+24.66$ °C. Ihre Spannweite betrug 40.46 °C. Die maximalen täglichen Regenmengen wurden bei Michelsrombach als 38.1 mm und in Schlitz als 47.8 mm gemessen. Im Untersuchungszeitraum lagen Tage ohne messbaren Niederschlag. In Tab. 4.2 (S. 83) sind die Mittelwerte für die in dieser Fallstudie relevanten unabhängigen Größen genannt. Die täglichen Abflussmittel als vorherzusagende Werte lagen zwischen 1 ℓ/s und dem festgelegten Höchstwert von 120 ℓ/s . Die Werte von Achsenabschnitten in Diagrammen sowie Intervallbreiten, Residuen und der RMSE bezüglich der [0,1]-Skala lassen sich damit in die Einheiten der ursprünglichen Skalen umrechnen (ggf. berücksichtige man dafür noch die angegebene Anzahl von Tagen).

Um zu vermeiden, dass die Generalisierungsfähigkeit der angewendeten Netzwerke durch die Auswahl zufällig oder abwechselnd gewählter Testdaten überschätzt wird, indem bei-

¹²⁾ Zum Beispiel bedeutet $u_2^{0-14}(t)$ die in Schlitz in den letzten beiden Wochen vor dem Zeitpunkt t gemessene Niederschlagsmenge. Die Summe der mittleren Tageslufttemperaturen an der am Breitenbach gelegenen Mess-Station „Haus B“ über 80 Tage vor dem t -ten Tag — einschließlich des Wertes für t — wird als $u_3^{0-80}(t)$ angegeben. Lässt sich $t_0 = 2318$ dem 15.08.1998 (TDS-Nr. 76) zuordnen, steht $u_4^{2-3}(t_0)$ für das dort beobachtete Tagesmittel des Breitenbach-Abflusses am 13.08.1998.

spielsweise ein überangepasstes Netz, das die Daten auswendig gelernt und einfach zwischen den sich wenig ändernden Werten des davor und danach liegenden Tages interpoliert hätte, dienten zum Testen des Modells und bei Kreuzvalidierungen jeweils zusammenhängende Zeitabschnitte. Außerdem konnten auf diese Weise für die Testdaten grafische Darstellungen der Zeitreihen erzeugt und Prognosen von Konkurrenzmodellen einfach berechnet werden.

Die Tage, an denen mindestens ein Messwert fehlte, wurden nicht für die Anpassung der Modellparameter berücksichtigt. Aus dem Zeitraum 01.01.1990–31.05.1998 lagen Ergebnisse von Messungen an 2242 Tagen vor, die zum Trainieren und Validieren von KNNs dienten. Auf diese wird im Folgenden als *Lerndaten* Bezug genommen. Zum Überprüfen der Prognosegüte wurden Werte an 945 aufeinander folgenden Tagen vom 01.06.1998 bis zum 31.12.2000 verwendet, welche die *Testdaten* bildeten.

In diesem Abschnitt können die mit *Testdatensatz-Nummer* (TDS-Nr.) beschrifteten Abszissen von Diagrammen als Zeitachse aufgefasst werden, denn jede dieser Nummern entspricht einem Tag, wobei dem 1. Juni 1998 die Nummer 1 und dem 31. Dezember 2000 die Nummer 945 zugeordnet ist.

Die Zeitreihen der Niederschläge waren im Testzeitraum vollständig, während in sieben Tagen in dieser Zeit keine Lufttemperatur- und in 29 Tagen keine Schüttungswerte vorlagen. Die Fehler der KNNs dafür gingen nicht in die Fehlerberechnungen ein.

4.2.3 Methoden und Software

Neben dem mit Hilfe der Software *Visualrbfn* (S. 35 ff.) simulierten RBFSOM-Netzwerk, dessen Aufbau, Funktionsweise und Möglichkeiten in Unterabschnitt 2.3.2 ab S. 26 vorgestellt werden, dienten LNNs und die *naive Prognose* $\check{y}(t) = y(t - 1)$ [Mic92, S. 140], auch *Persistenzmodell* genannt, als konkurrierende Modelle, die Mindestanforderungen an die globale Prognosegüte des RBFSOM-Netzes vorgaben.

Relevante Prädiktoren wurden mit GRNNs unter Verwendung der Software *Trajan* (die Bandbreite betrug 0.1) durch die Vorwärtsmethode (siehe S. 36) selektiert. Um den Zeitaufwand dafür im Rahmen von einigen Stunden zu halten, dienten von den Lerndaten 500 zufällig ausgewählte Datensätze zum Training und weitere 400 zum Validieren der Modellgüte.

Zur Benennung der Zellen im SOM-Gitter mit Angaben des jeweils ersten Datums, dessen zugehöriger Eingabevektor in das Einzugsgebiet des betreffenden SOM-Neurons fiel, etwa für die Darstellungen der U-Matrix und einer Sammon-Abbildung, wurde das Shell-Skript `label_cases.sh` in das Arbeitsverzeichnis kopiert und dort entsprechend modifiziert.

4.2.4 Ergebnisse

Zur Bildung eines Modells für die Beschreibung der statistischen Abhängigkeit täglicher Durchschnitte des Abflusses des Breitenbaches an der Probestelle „Haus B“, im Folgenden kurz als Y bezeichnet, von der dort gemessenen Lufttemperatur und den Nieder-

schlagsmengen über die oben genannten Zeiträume selektierten GRNNs sieben relevante Prädiktoren, die in Tab. 4.2 genannt sind.¹³⁾

Tabelle 4.2: Selektierte Prädiktoren für ein Regen-Temperatur-Abfluss-Modell des Breitenbaches. OW steht für den entsprechenden Messwert des $[0,1]$ -skalierten Merkmals in der ursprünglichen Skala. Aus der Bezeichnung ist der Zeitraum vor t in Tagen abzulesen, über den die Messwerte der genannten Größen summiert wurden. Mit Hilfe der angegebenen original-skalierten Minima (OMin) und Maxima (OMax) in Grad Celsius für die Lufttemperatur und Millimeter für die Niederschlagswerte lassen sich die im Folgenden genannten Werte der $[0,1]$ -Skala zurück transformieren.

Prädiktor	OW	Merkmal	Mess-Station	OMin	OMax
$X_1(t)$	$u_2^{0-7}(t)$	Niederschlag	Schlitz	0	136
$X_2(t)$	$u_2^{0-14}(t)$	Niederschlag	Schlitz	0	76.57
$X_3(t)$	$u_2^{14-28}(t)$	Niederschlag	Schlitz	0	85.14
$X_4(t)$	$u_1^{0-28}(t)$	Niederschlag	Michelsrombach	0.29	65.54
$X_5(t)$	$u_3^{0-28}(t)$	Lufttemperatur	„Haus B“	-9.27	20.87
$X_6(t)$	$u_3^{0-80}(t)$	Lufttemperatur	„Haus B“	-3.61	17.44
$X_7(t)$	$u_3^{80-160}(t)$	Lufttemperatur	„Haus B“	-3.61	17.44

Unter der Verwendung der Lerndaten wurden mit Hilfe von Kreuzvalidierungen die im folgenden genannten Parameter des RBF-SOM-Netzwerkes, vor allem die Ausmaße der rechteckig angeordneten SOM mit einer Breite von 13 und einer Höhe von 6 Zellen und damit eine Anzahl von 78 RBF-Zentren, optimiert. Die wichtigsten Optionen bei der Anwendung der Simulationssoftware *Visualrbfn* sind auf S. 123 wiedergegeben. Die Bandbreiten der RBFs wurden so gewählt, dass sie jeweils dem 2.5fachen der euklidischen Distanz des zugehörigen RBF-Zentrums zu dem zweitnächsten Nachbarn unter den RBF-Zentren entsprach.¹⁴⁾

Das Training der SOM wurde mit allen Lerndaten gemäß der Voreinstellungen des Shell-Skriptes `train_som.sh` mit einer Initialisierung durch Zufallszahlen, einer gaußschen Nachbarschaftsfunktion, einem linear auf eins abfallenden Nachbarschaftsradius mit einem der Breite der SOM entsprechenden Startwert sowie einer linear bis auf null abfallenden Lernrate von anfangs $\eta = 0.5$, vgl. Gl. 2.1 (S. 13) und [KHKL96, S. 13], nach 100'000 Lernschritten abgeschlossen.

Eine U-Matrix-Darstellung der trainierten SOM ist in Abb. 4.8 zu sehen. Jedes Element des SOM-Gitters ist mit dem Datum im Format JJMMTT — JJ beschreibt die letzten beiden Stellen der Jahreszahl, MM den Monat und TT den Tag im Monat — beschriftet,

¹³⁾ Um Doppelindizes bei den Zeitreihenelementen und deren Vektorkomponenten zu vermeiden, wurde die Schreibweise $\mathbf{x}(t)$ der in den vorangegangenen Kapiteln verwendeten \mathbf{x}_t vorgezogen.

¹⁴⁾ Die entsprechenden Zuweisungen in der Datei `options.m` sind `RBFBandWidthNN = 2` und `RBFBandWidthStretch = 2.5`.

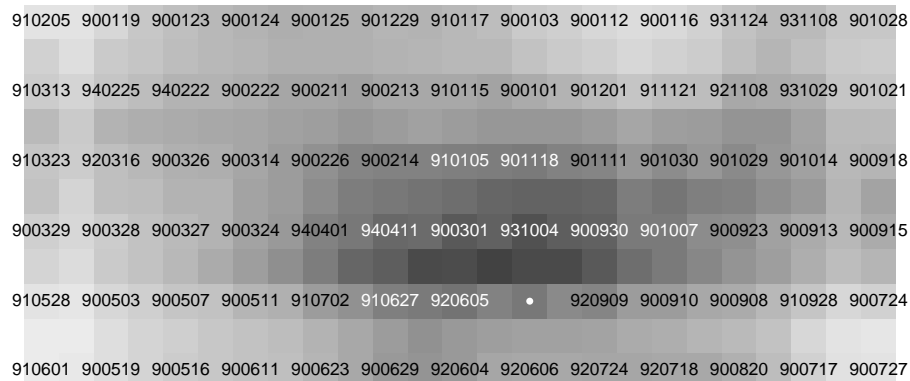


Abbildung 4.8: U-Matrix-Darstellung der verwendeten 13×6 -SOM auf der Basis der in der Fallstudie ausgewählten Wetter- und Abfluss-Daten. Dunkle Graustufen bedeuten große Abstände zwischen den benachbarten Kodebuchvektoren. Die Beschriftung der Zellen gibt das Datum (Jahr, Monat, Tag; vgl. Text) des ersten Trainingsdatensatzes an, der in das Einzugsgebiet des betreffenden SOM-Neurons fiel.

dem der jeweils erste Lerndatenvektor im Einzugsgebiet des betreffenden SOM-Neurons zugeordnet ist.¹⁵⁾

Abb. 4.9 zeigt eine Sammon-Abbildung der SOM-Kodebuchvektoren, die mit den genannten Beschriftungen versehen sind. Die Anordnung ist im Vergleich zur U-Matrix in Abb. 4.8, die sonst für alle folgenden Darstellungen der SOM maßgeblich für die Anordnung der SOM-Neuronen ist, an einer horizontalen Achse gespiegelt.

Den in der U-Matrix-Darstellung dunklen Bereichen entsprechen in der Sammon-Abbildung große Distanzen, die wiederum große Abstände der zugehörigen Kodebuchvektoren im Eingaberaum andeuten. Das SOM-Neuron mit der Nummer 60, dessen Gitterzelle in Abb. 4.8 in der achten Spalte von links und der zweiten Zeile von unten durch einen weißen Punkt markiert ist und dessen Kodebuchvektor-Abbild in Abb. 4.9 keine Beschriftung trägt, war für keinen Lerndatensatz Sieger, d. h. sein Einzugsgebiet enthielt keine für das Training verwendeten Eingabedaten.

Die Verteilungen der als Graustufen abgebildeten Werte der sieben Komponenten $x_j(t)$ ($j = 1, \dots, 7$) der SOM-Kodebuchvektoren veranschaulicht Abb. 4.10. Die Darstellung verdeutlicht, dass die Werte der Lufttemperaturen für die 28 und 80 Tage vor dem Zeitpunkt t (Merkmale X_5 und X_6) unten rechts am höchsten, oben links am niedrigsten und sonst mittelmäßig ausfallen. Bei der zusätzlichen Betrachtung der Werteverteilung der Größe X_7 lässt sich feststellen, dass die SOM Daten aus dem Winter oben links, aus dem Sommer unten rechts, aus dem Herbst oben rechts (hohe Temperaturen 80 Tage und früher vor t) und aus dem Frühling unten links abbildet. Die Beschriftungen der SOM-Gitterelemente in der Darstellung der U-Matrix stützen diese Aussage. Dabei lässt sich

¹⁵⁾ Zum Beispiel steht der Kodewert 980815 für das Datum 15.08.1998. Die Zuordnung ist für den Untersuchungszeitraum auch mit der zweistelligen Angabe der Jahreszahl eindeutig.

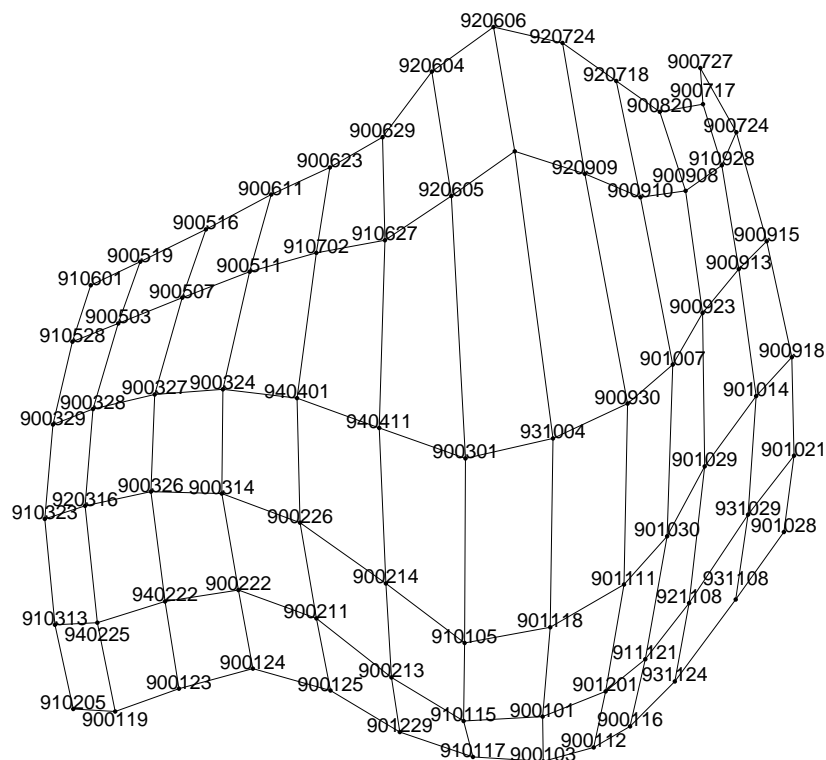


Abbildung 4.9: Sammon-Abbildung der SOM-Kodebuchvektoren, vgl. Abb. 4.8.

aus den Diagrammen der vier positiv korrelierten Niederschlagsmerkmale X_1 , X_2 , X_3 und X_4 schließen, dass den trockenen Perioden Randpositionen zugewiesen sind.¹⁶⁾

Abb. 4.11 zeigt die Muster der Prototypen für jede Klasse von Eingabedaten. Sie kann die Veranschaulichung der Verteilung der Eingabewerte auf der SOM ergänzen.

Bei der Übersicht über die [0,1]-skalierten Kodebuchvektoren, in Abb. 4.12 als Balkendiagramme, wird noch deutlicher, dass links oben im SOM-Gitter die niedrigsten Niederschlags- und Temperaturwerte in den Zeitabschnitten unmittelbar vor dem Prognosezeitpunkt liegen. In der Mitte sind die Daten und ihre Prototypen positioniert, welche die höchsten Niederschlagssummen aufweisen.

Für Zeitsequenzen ist es aufschlussreich, die SOM-Abbildungen der Eingabemuster auf dem SOM-Gitter zu aufeinander folgenden Zeitpunkten als sog. *Trajektorien* zu veranschaulichen. Abb. 4.13 zeigt beispielhaft die Folge der Siegerneuronen für die Eingabemuster $x(t)$ der 31 Tage im Dezember 1999 mit den Testdatensatz-Nummern 549 bis 579. Die den Siegerneuronen entsprechenden Gitterelemente wurden nummeriert.¹⁷⁾ Die mit diesen Nummern beschriebene Folge von Siegerneuronen ist (beginnend mit dem

¹⁶⁾ Bei Vorstudien, bei denen die Größe X_7 nicht verwendet wurde, bildeten sich SOMs mit sehr gleichmäßigen Kodebuchvektor-Abständen im Eingaberaum und einer intrinsischen Dimensionalität von 2. Die feuchten kühlen, die feuchten warmen, die trockenen kühlen und die trockenen warmen Perioden belegten je eine Ecke der SOM-Gitter.

¹⁷⁾ Die ursprünglich von `planes(.exe)` erzeugte PostScript-Datei wurde mit einem `Awk`-Skript des Verfassers nachbearbeitet.

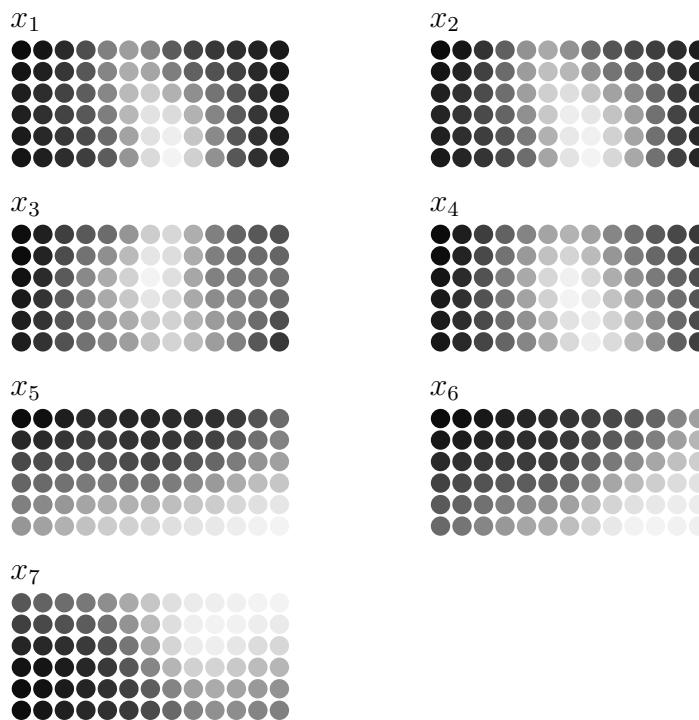


Abbildung 4.10: Auf dem SOM-Gitter dargestellte Werte der Komponenten der Kodebuchvektoren (auf der Basis der Lerndaten). Helle Graustufen deuten hohe Werte an. Je dunkler ein Grafikelement dargestellt ist, desto niedriger ist der zugehörige Wert des Merkmals X_j . Die Anordnung der SOM-Neuronen in den Darstellungen entspricht der U-Matrix in Abb. 4.8.

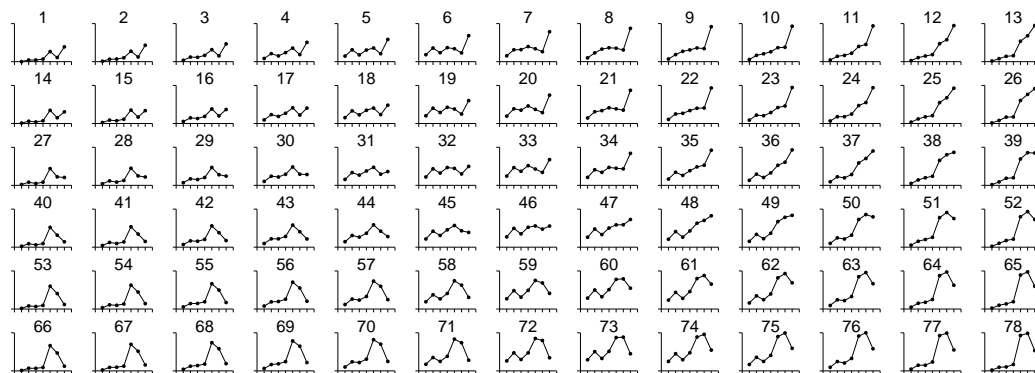


Abbildung 4.11: Darstellung der Kodebuchvektoren \mathbf{z}_k ($k = 1, \dots, 78$) der SOM in der entsprechenden Gitterstruktur. In jedem einzelnen Diagramm repräsentiert der Punkt j (von links) den Wert der Kodebuchvektorkomponente für die in Tab. 4.2 (S. 83) genannten Merkmale X_j , wobei die Ordinaten die reskalierten Werte im Intervall $[0, 1]$ beschreiben.

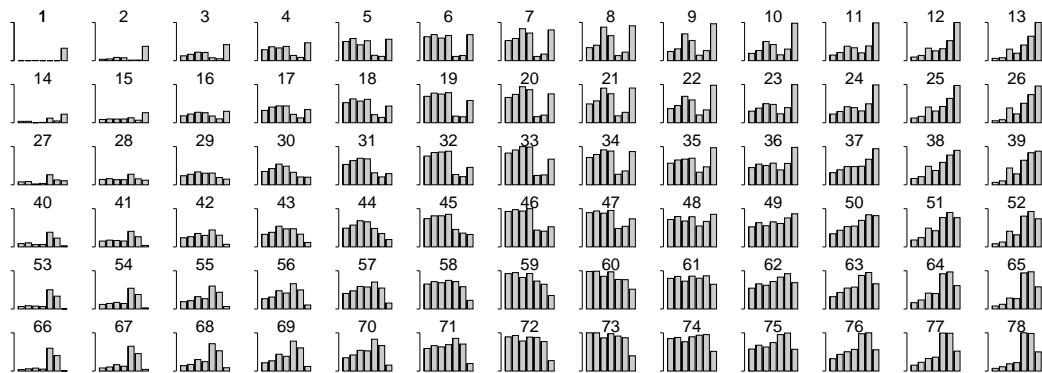


Abbildung 4.12: Darstellung der (erneut) $[0,1]$ -reskalierten Kodebuchvektoren als Balkendiagramme in der Anordnung des SOM-Gitters, vgl. Abb. 4.11.

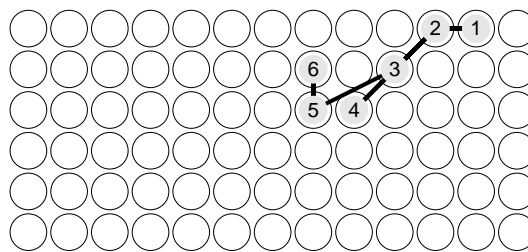


Abbildung 4.13: Trajektorie der sechs Siegerneuronen, auf welche die den 31 Tagen im Dezember 1999 entsprechenden Netzeingaben abgebildet wurden.

01.12.1999) 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 3, 4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 3, 3, 5, 5, 6, 5, 5, 6.
 Abb. 4.14 zeigt die Eingabedaten für den jeweils ersten Tag eines jeden Monats im Jahr 2000. Die jahreszeitlich bedingte Struktur der SOM wird auch hier deutlich. In diesem

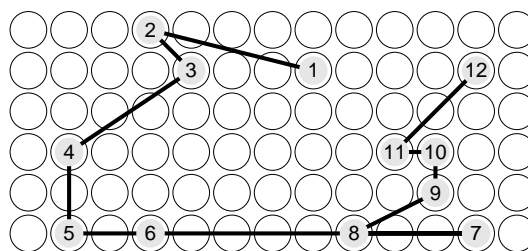


Abbildung 4.14: Trajektorie der SOM-Sieger in der verdeckten Schicht des RBFSOM-Netzwerkes für die in Tab. 4.2 genannten Eingabemuster des jeweils ersten Tages eines Monats ($1 \cong 1.$ Januar, $2 \cong 1.$ Februar, \dots , $12 \cong 1.$ Dezember) im Jahr 2000.

Fall wird die Karte entgegen dem Uhrzeigersinn durchlaufen. Bei der Beurteilung der Stärke der Veränderung von einem zum nächsten Monat — besonders vom 1. Juni zum 1. Juli (Sprungstellen 6 und 7) oder 1. Oktober zum 1. November (Sprungstellen 10 und

11) — ist nicht nur die Länge eines Sprunges auf dieser Karte zugrunde zu legen, sondern zusätzlich die durch die Sammon-Abbildung in Abb. 4.9 oder die durch Graustufen ausgedrückten Entfernungen auf der U-Matrix in Abb. 4.8 zu berücksichtigen. Die Darstellung trifft keine Aussage über die ökologische Wirksamkeit der monatlichen Veränderungen. Die Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$ von Lerndaten, die in den Einzugsgebieten der einzelnen SOM-Neuronen liegen, lässt sich durch Zahlenangaben und Graustufen auf dem SOM-Gitter in der verdeckten Schicht des RBFSOM-Netzes veranschaulichen, wie Abb. 4.15 zeigt.

69	50	41	34	21	44	20	39	49	73	30	22	26
30	6	17	11	13	14	7	5	30	20	15	3	28
73	10	32	33	28	17	47	21	31	43	15	23	37
40	28	30	15	25	11	2	2	21	29	18	32	16
48	6	18	11	4	5	1	0	2	13	36	9	19
43	27	58	38	66	46	26	62	41	52	55	41	119

Abbildung 4.15: SOM-Gitter mit der für jede Zelle durch Graustufen und eine Aufschrift angegebenen Anzahl von Lerndaten, die in die jeweiligen Einzugsgebiete der SOM-Neuronen fallen ($\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$).

Hier hat die verhältnismäßig große Anzahl von 119 Lerndatensätzen, die in das Einzugsgebiet des Neurons fallen, das im Gitter unten rechts liegt, ein sehr starkes Gewicht. Dadurch erscheinen die Graustufen der restlichen Elemente dunkel.

Informationen über die geschätzten Dichten der Datenpunkte in Umgebungen der Kodebuchvektoren sind in Abb. 4.16 gegeben. Die Likelihoods des Parzen-Dichteschätzers und des im VI-Netz enthaltenen Ansatzes zur Bestimmung des Extrapolationsgrades werden unterschiedlich ermittelt, aber die Darstellungen haben eine große Ähnlichkeit.



Abbildung 4.16: Graustufendarstellung auf dem SOM-Gitter $[0,1]$ -skalierter Likelihoods des Parzen-Dichteschätzers (links) und des VI-Netzes für die Kodebuchvektoren (rechts).

Die beiden Darstellungen der Likelihoods in Abb. 4.16 haben gegenüber derjenigen in Abb. 4.15 einen stark geglätteten Charakter. In dem dunklen Bereich mit niedriger Dichte im Zentrum der Diagramme liegen die Kodebuchvektoren (wie in der U-Matrix und der Sammon-Abbildung zu sehen ist) wesentlich weiter voneinander entfernt als in anderen Bereichen der Datenpunktwolke. Dadurch ist die Anzahl der Datenpunkte pro Volumen geringer. Hingegen spielen die Volumina der SOM-Neuronen-Einzugsgebiete keine Rolle für die oben abgebildeten Graustufen. Weiterhin sind in den letzten beiden Abbildungen

die Likelihoods für die Kodebuchvektoren und nicht über die ganzen Einzugsgebiete der SOM-Neuronen angeben.

Betrachtet man im Folgenden auch die Werte $y(t)$ der abhängigen Größe Y , lassen sich auf den Elementen des SOM-Gitters durch Graustufen Streuungsmaße der Werte $y(t)$ abbilden, deren zugehörige Eingabemuster $\mathbf{x}(t)$ in die Einzugsgebiete der entsprechenden SOM-Neuronen fallen. Die linke Grafik in Abb. 4.17 zeigt dies am Beispiel von Spannweiten $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L) - \min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$. Für das SOM-Neuron mit bezüglich der Lerndatenmuster leerem Einzugsgebiet ist die Streuung der Ausgabegröße nicht definiert.



Abbildung 4.17: Spannweiten $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L) - \min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ der Ausgabegröße Y (tägl. Abfluss des Breitenbaches an „Haus B“) für die zugehörigen Einzugsgebiete der SOM-Neuronen in den Lerndaten (links; anstatt durch die voreingestellte Beschriftung „ndef“ wurde hier wegen der verkleinerten Grafik ein X zur Kennzeichnung nicht definierter Werte verwendet) und durch Graustufen repräsentierte Ausgaben des RBFNs bei der Eingabe der jedem SOM-Neuron zugeordneten Kodebuchvektoren (rechts).

Die rechte Grafik in Abb. 4.17 zeigt Ausgaben des RBFNs für die als Netzeingaben dienenden Kodebuchvektoren. Unter Berücksichtigung der Werte der Komponenten der Eingabevektoren aus Abb. 4.10 (S. 86) sind die durch helle Bereiche gekennzeichneten größten Abflusswerte nach hohen Niederschlägen zu erwarten, die mit mittleren Temperaturen zusammenfallen.

Die Abschätzung der lokal zu erwartenden Prognosefehler basierte nicht — wie dies einfacher und schneller zu realisieren, aber aufgrund der Anpassung an die Daten nicht mehr repräsentativ gewesen wäre — auf den Residuen der Lerndaten, sondern auf einer Kreuzvalidierung. Dabei dienten vier Validierungsdatensätzen mit je 450 und einer mit 442 Datensätzen (ihre Nummern waren jeweils aufeinander folgend, da sonst die teilweise stark autoregressiven Einflüsse die Schätzung des RMSEs mit einem systematischen Fehler versehen hätten) zum Berechnen der Validierungsfehler. Die restlichen 1792 bzw. 1800 Datensätze wurden zum Trainieren verwendet. In jedem Fall wurde ein neues RBFSOM-Netzwerk gebildet. Allein die Anzahl von RBF-Zentren und die Anzahl der nächsten Nachbarn zur Bandbreitenbestimmung waren konstant. Die mit CV beginnenden Kreuzvalidierungsoptionen und die benutzerdefinierten Werte sind auf S. 123 wiedergegeben. Bei den fünf Kreuzvalidierungen lagen die Wurzeln der mittleren Fehlerquadratsummen für die Validierungsdaten im Mittel bei 0.1755. Ihre Standardabweichung war 0.0128, der kleinste dieser Werte betrug 0.1618 und der Maximalwert lag bei 0.1936.

Mit den für jeden als Validierungsdatensatz verwendeten Lerndatensatz vorliegenden Residuen ließen sich Fehlerbalken und die Abschätzungen der erwarteten Fehler durch das

VI-Netz für alle Kodebuchvektoren berechnen und grafisch als Graustufen auf dem SOM-Gitter darstellen, siehe Abb. 4.18. Auch hier liegt eine gegenüber der auf der SOM basierenden Ausgabe ähnliche, aber glattere Darstellung vor.



Abbildung 4.18: Durch Graustufen kodierte, $[0,1]$ -skalierte und auf dem SOM-Gitter dargestellte Konfidenz- (links) und Prognoseintervallbreiten (rechts) eines VI-Netzes für die Kodebuchvektoren aufgrund von Kreuzvalidierungsdaten.

Aus den Grafiken wird deutlich, dass große Abflussprognosen große Fehler und Streuungen erwarten lassen.

Bei den Prognosen der täglichen $[0,1]$ -skalierten Abflusswerte des Breitenbaches für die Testdaten betragen die Wurzel der mittleren Fehlerquadratsumme $E_{\text{RMS}} = 0.1057$ und das Bestimmtheitsmaß $B = 0.3439$.

Die folgenden Abbildungen 4.19 bis 4.31 beinhalten eine Veranschaulichung der in der Datei `work_results.txt`, vgl. Tab. 3.3 auf S. 58, gespeicherten Ergebnisse der Prognosen des RBFSOM-Netzes für die Testdaten.

Die Messwerte sind den Netzwerkprognosen in Abb. 4.19 in einer Zeitreihendarstellung und einem Streudiagramm gegenüber gestellt.

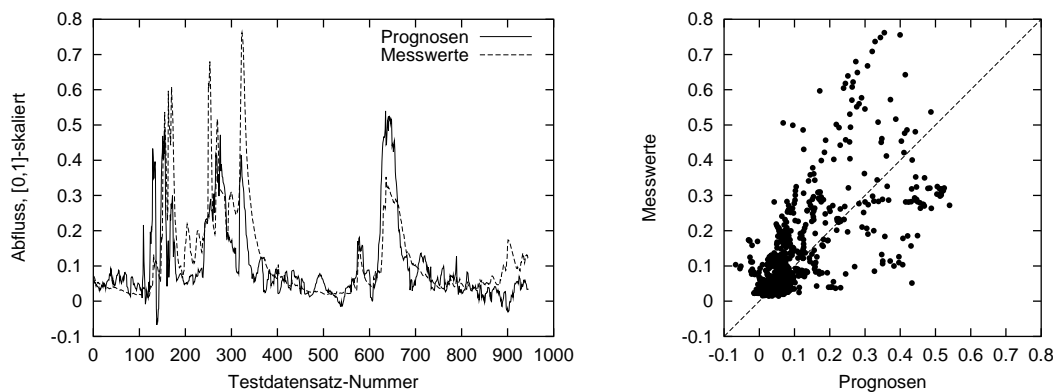


Abbildung 4.19: Messwerte und Prognosen des RBFSOM-Netzwerkes für die Testdaten als Zeitreihendarstellung (links) und Streudiagramm (rechts). Eine Einheit auf der Ordinatenachse entspricht in der ursprünglichen Skala $119 \ell/s$. Die Testdatensatz-Nummern zählen die Tage ab dem 01.06.1998; die Abszisse des linken Diagramms kann daher als Zeitachse aufgefasst werden.

Negative Prognosewerte sind in dieser Anwendung prinzipiell nicht unplausibel, solange der in die ursprüngliche Skala zurück transformierte Wert für den Abfluss des Breitenbaches positiv bleibt, denn es ist äußerst unwahrscheinlich, dass der Bach an der Messstation „Haus B“ trocken fällt.

Das mehr oder weniger langsame Abfließen ist mit diesem Modell schlecht abgebildet, da für jede einzelne Prognose keine Informationen über vorhergehende Abflusswerte vorlagen.

Ein in Abb. 4.20 gezeigtes Histogramm der Residuen $y(t) - \hat{y}(t)$ dient der Veranschaulichung ihrer Verteilung. Ein statistischer Test auf das Vorliegen einer Normalverteilung der Residuen wurde an dieser Stelle nicht durchgeführt.

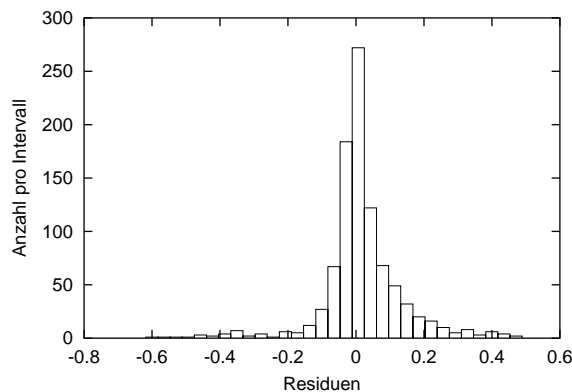


Abbildung 4.20: Histogramm der Residuen bei der Prognose des täglichen Abflusses des Breitenbaches an „Haus B“ für die Testdaten mit einem RBFSOM-Netzwerk.

Die Residuen für jeden Testdatensatz sind jahreszeitlich verschieden, wie Abb. 4.21 zeigt. Beispielsweise ergeben sich für die Spätsommer niedrige absolute Prognosefehler (z. B. fällt der 1. August auf die Testdatensatz-Nummern 62, 427 und 793).

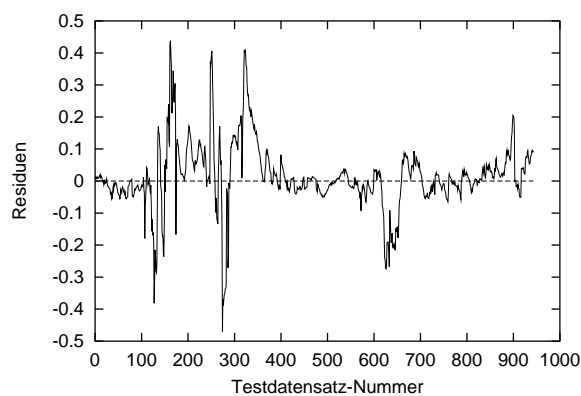


Abbildung 4.21: Residuen bei der Prognose des täglichen Abflusses des Breitenbaches an „Haus B“ für die einzelnen Testdatensätze.

Die Residuen sind von der Höhe der Vorhersage abhängig und es liegt keine *Homoskedastizität* vor, siehe Abb. 4.22. Bei hohen Prognosewerten sind auch (betragsmäßig) größere Fehler zu erwarten.

Um diese Unterschiede zwischen den gemessenen Werten und den Prognosen des RBFSOM-Modells zu erklären, können die euklidischen Abstände zwischen den Testeingabe-

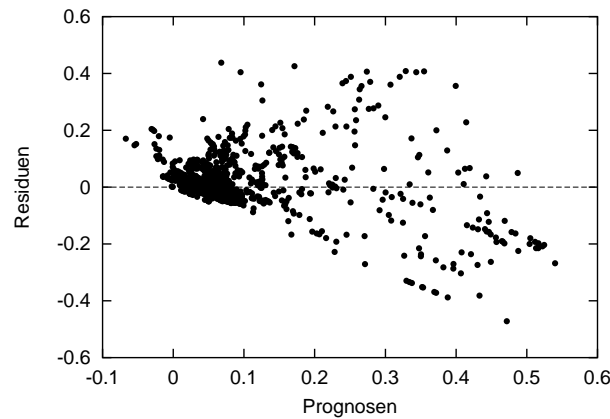


Abbildung 4.22: *Abhängigkeit der Residuen von den Werten der Prognosen für die Testdaten.*

vektoren und den jeweils nächstliegenden Kodebuchvektoren bzw. RBF-Zentren (Quantisierungsfehler) Hinweise liefern. Dieses einfache Maß für den Grad der Unbekanntheit von Eingabevektoren ist in Abb. 4.23 für jeden Testdatensatz dargestellt.

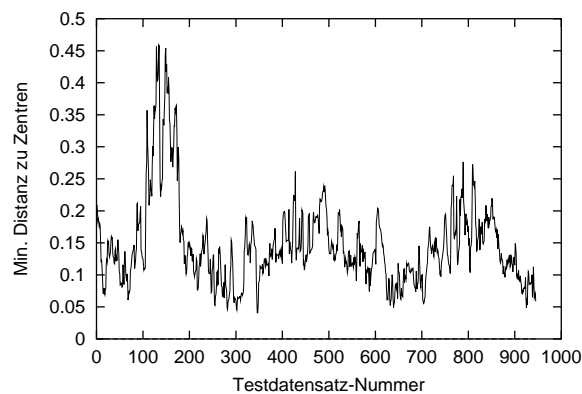


Abbildung 4.23: *Minimale euklidische Abstände der SOM-Kodebuchvektoren zu den Netzeingabevektoren für alle Testdaten (Zeitreihendarstellung).*

Die maximale Aktivität aller RBF-Neuronen bei der Eingabe eines Vektors in das Netzwerk ist ein weiterer Indikator für den Grad der Extrapolation. Dieser ist für jeden Testdatensatz in Abb. 4.24 dargestellt. Viele der Testdatensätze mit den Nummern zwischen 100 und 200 sind verhältnismäßig wenig bekannt.

Hier ist eine starke negative Korrelation — der pearsonsche Korrelationskoeffizient beträgt -0.8796 — zwischen beiden Größen offensichtlich. Beide Abbildungen zeigen, dass die größten Aktivitäten der RBF-Neuronen (meistens) für die Eingaben zu erwarten sind, die den kleinsten euklidischen Abstand zum nächsten Kodebuchvektor aufweisen. Umgekehrt hängt eine große Minimal-Distanz zwischen Eingabe- und Kodebuchvektoren mit niedrigen Maximalwerten der RBFs zusammen. Die Werte der RBFs hängen per definitionem von den Abständen der Kodebuchvektoren z_k zu den Eingaben x ab. Die Verwen-

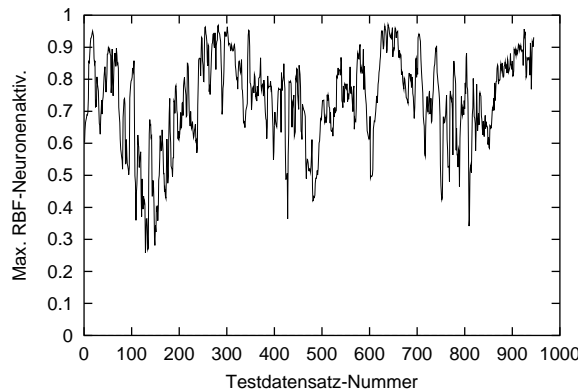


Abbildung 4.24: Zeitreihendarstellung der maximalen Aktivitäten der RBF-Neuronen in Abhängigkeit von den als Eingabe in das RBF-SOM-Netzwerk dienenden Testdatensätzen.

dung gaußscher Basisfunktionen gemäß Gl. 2.3 (S. 17) und die unterschiedlich großen Bandbreiten der einzelnen RBFs sind dafür verantwortlich, dass der pearsonsche Korrelationskoeffizient den Wert -1 nicht annehmen kann.¹⁸⁾

Allerdings verdeutlicht Abb. 4.25, dass zwar bei guter „Bekanntheit“, was aufgrund geringer Abstände der Eingabevektoren zu den Kodebuchvektoren bzw. RBF-Zentren angenommen werden kann, in vielen Fällen geringe Fehler bei der Prognose vorliegen, dass für die Testdaten aber nicht grundsätzlich von einer großen Distanz der Eingabevektoren zu den RBF-Zentren auf betragsmäßig große Prognose-Fehler geschlossen werden kann.

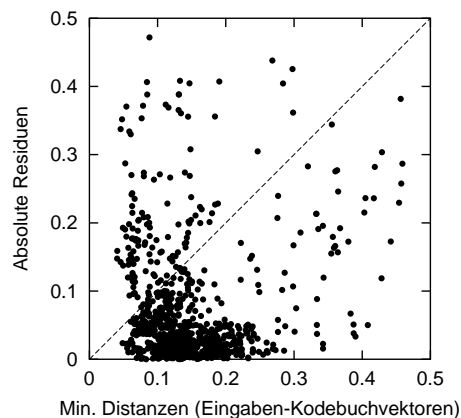


Abbildung 4.25: Beträge der Residuen der Prognosen über den minimalen euklidischen Abständen der Testeingabevektoren zu den Kodebuchvektoren (RBF-Zentren).

Die Anzahl $\#\mathcal{X}_{\kappa_i}^L(\mathbf{x}(t))$ von Lerndaten, die durch ihre Zugehörigkeit zu einem Einzugsgebiet die Netzwerkprognosen in diesem Bereich stützen könnten, ist in Abb. 4.26 zu sehen.

¹⁸⁾ Der entsprechende pearsonsche Korrelationskoeffizient liegt für das in der Fallstudie verwendete RBF-SOM-Netz, wenn konstante Bandbreiten der RBFs von 0.4 verwendet werden, bei -0.9952 .

Viele der Prognosen für Testdaten mit den Nummern zwischen etwa 100 und 160 sind nur schwach durch Lerndaten gestützt.

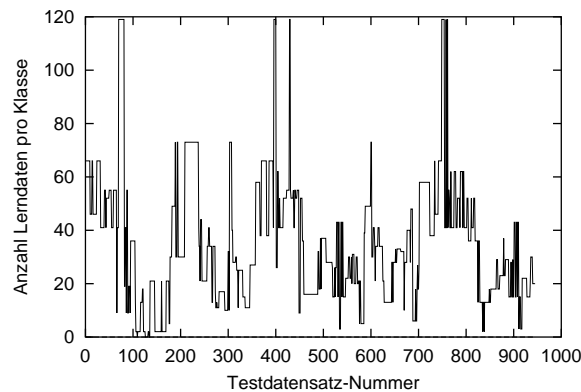


Abbildung 4.26: Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$ von Eingabevektoren unter den Lerndaten, die in dieselben Einzugsgebiete der SOM-Neuronen wie die Test-Eingabedaten fallen, dargestellt für alle Testdaten.

Bei der Betrachtung der Spannweiten der $y(t)$ in den einzelnen SOM-Klassen in Abb. 4.27 zeigt sich in der Tendenz für die Testdaten eine Saisonalität mit einer niedrigeren Streuung im Sommer. An einigen Stellen sind die Spannweiten sehr groß.

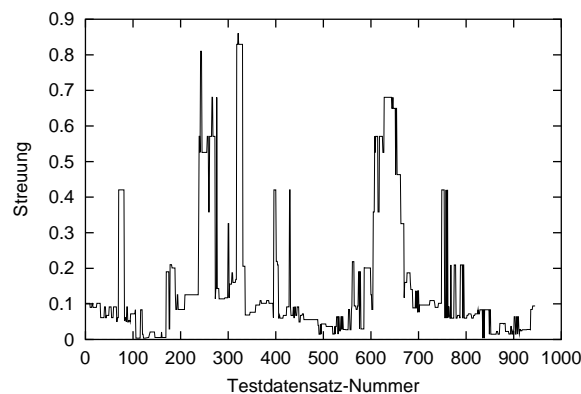


Abbildung 4.27: Spannweite zwischen den Werten der Ausgabegröße in den einzelnen SOM-Klassen.

Abb. 4.28 macht deutlich, dass die Prognose von Y durch die arithmetischen Mittel $\bar{y}_{\kappa(\mathbf{x}(t))}$ der Elemente von $\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L$ mit $E_{\text{RMS}} = 0.1123$ eine Konkurrenz zu denen des RBF-SOM-Netzes ($E_{\text{RMS}} = 0.1057$, siehe oben) darstellt. Die beste, schrittweise (vgl. S. 36) mit SPSS gefundene lineare Regressionsfunktion lieferte ohne die Notwendigkeit einer vorigen Skalentransformation ein einfach zu interpretierendes Modell, das auf 20 Prädiktoren beruhte und dessen auf die $[0,1]$ -Skala umgerechneter RMSE für die Testdaten bei 0.1361 lag.

Schätzungen für Prognoseintervall-Grenzen für die Ausgabegröße in den Testdaten sind durch ihre Minima und Maxima innerhalb der SOM-Klassen bezüglich der Lerndaten

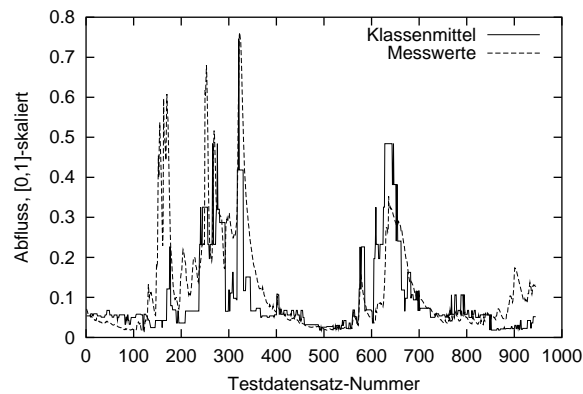


Abbildung 4.28: Prognose von Y durch die arithmetischen Mittel $\bar{y}_{\kappa(\mathbf{x}(t))}$ über die Elemente von $\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L$.

gegeben. In Abb. 4.29 liegen die gemessenen Testdaten der abhängigen Größe weitgehend in diesem Intervall.¹⁹⁾

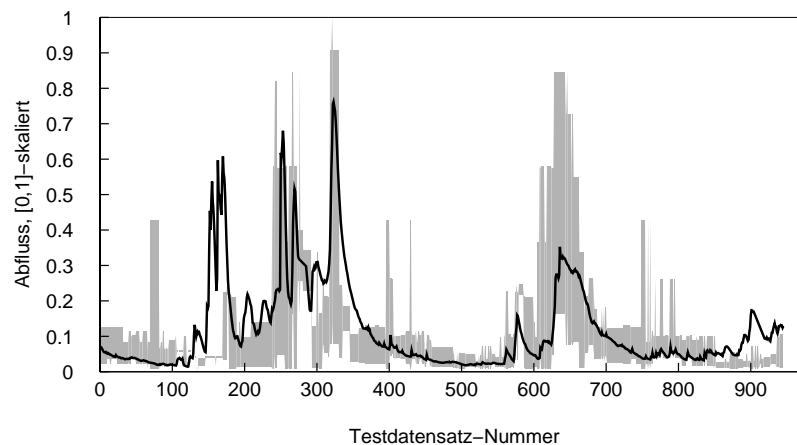


Abbildung 4.29: Testdaten (schwarz) mit den Minima $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und Maxima $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ der jeweiligen Klassen, die durch die SOM auf der Basis der Lern-Eingabedaten bestimmt sind, als einfache Schätzung eines Prognoseintervalls (grau).

Die Testdaten befinden sich ebenfalls weitgehend innerhalb der um die Prognosen gezeichneten Fehlerbalken (Predictive-Error-Bars), die durch die oben beschriebene Kreuzvalidierung bestimmt wurden, siehe Abb. 4.30.

Vergleicht man die Darstellungen 4.15 und 4.31, zeigt sich eine andere Verteilung für die Testdaten als für die zur Modellanpassung verwendeten Daten auf dem SOM-Gitter.

¹⁹⁾ Diese Grafik wurde nicht automatisch von *Visualrbfn* erzeugt. Stattdessen wurde ein Shell-Skript verfasst, das eine *Ploticus*-Steuerdatei generierte. Dieses Skript kann in zukünftige Versionen von *Visualrbfn* eingebunden werden.

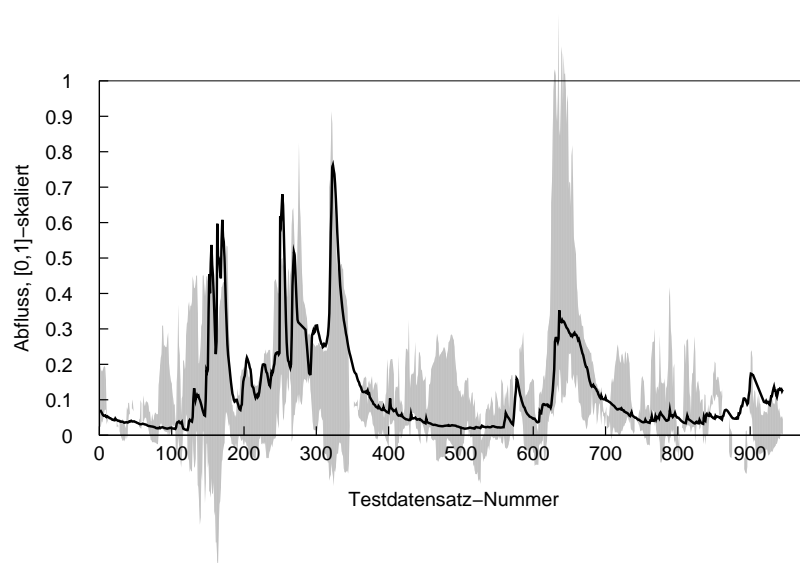


Abbildung 4.30: Testdaten (schwarz) der abhängigen Größe mit den um die Prognosewerte gezeichneten Fehlerbalken (Predictive-Error-Bars, grau).

0	0	9	28	16	3	0	2	19	34	31	20	4
0	0	13	6	20	0	0	10	4	13	15	7	15
5	9	9	9	9	0	0	10	4	16	18	11	12
9	12	2	5	12	1	0	25	11	11	23	2	30
4	8	4	10	0	0	0	9	15	26	28	8	5
0	14	31	18	47	19	3	14	33	39	33	26	27

Abbildung 4.31: Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^T$ von Testdaten, die in die jeweiligen Einzugsgebiete der SOM-Neuronen fallen.

Eine Abschätzung des lokal zu erwartenden Fehlers ist auch durch den RMSE gegeben, der für die einzelnen SOM-Klassen auf der Basis der Testdaten berechnet wurde. Abb. 4.32 zeigt zudem die Anzahl der Testdaten, die der Berechnung dieses lokal erwarteten RMSE jeweils zugrunde liegen.

Der lokale RMSE auf den Testdaten schwankt — wo er berechenbar ist — zwischen 0.00646 und 0.322917. Vor allem für die SOM-Neuronen-Einzugsgebiete, die in Abb. 4.32 unten rechts liegen (vgl. die Darstellungen der Werte der Komponenten in Abb. 4.10) ist ein kleiner RMSE zu beobachten. Zudem ist dort die Streuung gering. Gleichzeitig zeigen die Beschriftungen der einzelnen SOM-Zellen an, dass sich die Angaben auf verhältnismäßig viele Datensätze stützen. Daher kann man in Netzprognosen für den Spätsommer nach längeren Trockenperioden vergleichsweise großes Vertrauen setzen.

Bei der Analyse des Netzwerkverhaltens und der Abschätzung der Vertrauenswürdigkeit einzelner Netzwerkausgaben sind insbesondere die im Folgenden dargestellten Möglichkeiten wichtig. Dabei können die Aktivierungen der RBF-Neuronen und die Abstände von den Prototypen auf dem SOM-Gitter für einen beliebigen Eingabevektor, zunächst

ndef	ndef	9	28	16	3	ndef	2	19	34	31	20	4
ndef	ndef	13	6	20	ndef	ndef	10	4	13	15	7	15
5	9	9	9	9	ndef	ndef	10	4	16	18	11	12
9	12	2	5	12	1	ndef	25	11	11	23	2	30
4	8	4	10	ndef	ndef	ndef	9	15	26	28	8	5
ndef	14	31	18	47	19	3	14	33	39	33	26	27

Abbildung 4.32: Darstellung des SOM-Gitters mit Graustufen der für jedes SOM-Neuronen-Einzugsgebiet zu erwartenden $[0,1]$ -skalierten Fehlermaße (RMSE) und mit den Anzahlen von Testdaten, die in das Einzugsgebiet des jeweiligen SOM-Neurons fallen, als Beschriftungen. Die Zellen, für die keine Testdaten im Einzugsgebiet des betreffenden SOM-Neurons liegen und für die der lokale RMSE nicht definiert ist, sind durch „ndef“ gekennzeichnet.

am Beispiel ausgewählter Testdatenvektoren, als Graustufen dargestellt werden. In dieser Fallstudie wurden für die 945 Testdaten 2835 EPS-Dateien der absoluten und relativierten RBF-Aktivitäten sowie der euklidischen Distanzen erzeugt.

Abb. 4.33 veranschaulicht den Grad der Bekanntheit der Testdaten-Eingabevektoren mit den Nummern 129 und 272 als zwei extreme Fälle durch deren euklidischen Abstände zu den Kodebuchvektoren der SOM.

TDS-Nr. 129 (07.10.1998)



TDS-Nr. 272 (27.02.1999)

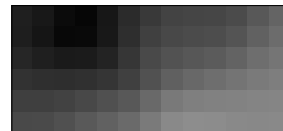


Abbildung 4.33: Durch Graustufen auf dem SOM-Gitter dargestellte relativierte euklidische Abstände der Kodebuchvektoren und der Netzeingabevektoren mit den Testdatensatz-Nummern 129 und 272. Die Graustufen sind so abgestimmt, dass Schwarz den kleinsten und Weiß den größten Abstand zwischen allen Test-Eingabedaten und allen Kodebuchvektoren ausdrückt.

Mit Hilfe der Darstellungen der RBF-Neuronenaktivitäten wird das Zustandekommen einer Netzausgabe in Abb. 4.34 deutlicher als durch die vorangegangene Grafik mit euklidischen Distanzen. Für jede dieser Grafiken sind die RBF-Neuronen-Aktivitäten Graustufen durch eine affine Funktion zugeordnet, sodass — ausnahmsweise — schwarze Zellen die Neuronen mit der größten Aktivität anzeigen und die geringsten Erregungen von Neuronen der verdeckten Schicht des RBFNs durch weiße Grafikelemente dargestellt werden. Es handelt sich dabei um „*Negative*“ mit invertierten Graustufen.²⁰⁾

²⁰⁾ Auf diese Weise sind die wenigen dunklen Bereiche besser erkennbar. Vor allem die Darstellung der RBF-Neuronenaktivitäten bei der Eingabe der Test-Eingabedaten mit der Nummer 129 spricht für die Verwendung der *visusom*-Option `-framed 1`.

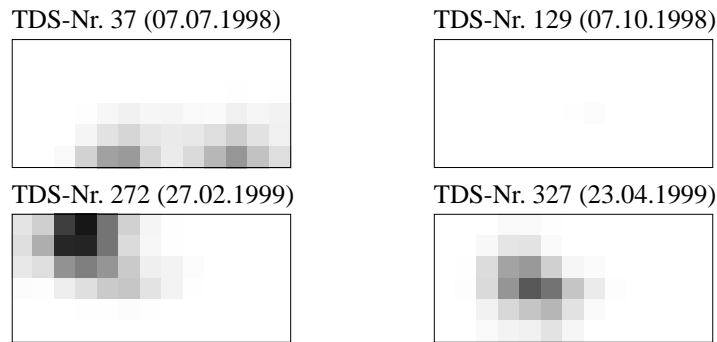


Abbildung 4.34: Aktivitäten der einzelnen RBF-Neuronen des RBFSOM-Netzes für die genannten Eingaben als relativierte Graustufen, wobei hier „Negative“ (invertierte Graustufen) dargestellt sind.

Neben als weitgehend unbekannt zu bezeichnenden Eingaben, wie im Fall des Testdatensatzes mit der Nummer 129, und einige Neuronen sehr stark aktivierenden Eingaben, z. B. der Testdatenvektor mit der Nummer 272, gibt es Netzeingaben, die eine mäßige Erregung erzeugen (TDS-Nr. 327) und auch solche, die auf dem SOM-Gitter weit auseinander liegende RBF-Zentren erregen (TDS-Nr. 37).

Die Betrachtung der relativierten Darstellungen, bei denen die für jeden einzelnen Eingabevektor größte Aktivität der RBF-Neuronen schwarz und die kleinste Erregung weiß wiedergegeben wird, kann die Neuronen mit der höchsten Erregung verdeutlichen, wenn, wie im Beispiel des Testdatensatzes mit der Nummer 129, in der vorhergehenden Darstellung kaum etwas zu erkennen ist. Ein Beispiel ist in Abb. 4.35 gegeben.

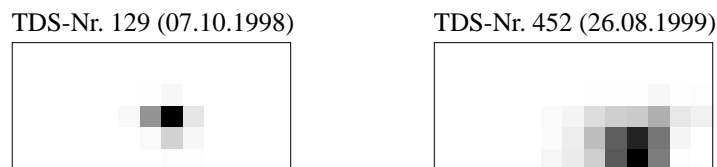


Abbildung 4.35: Relativierte Aktivitäten der auf das SOM-Gitter abgebildeten RBF-Neuronen-Aktivitäten für zwei ausgewählte Eingabevektoren aus den Testdaten als Graustufenbild („Negativdarstellung“).

Neben wirklich gemessenen Werten aus den Testdaten lassen sich auch fiktive Werte in das RBFSOM-Netz eingeben, beispielsweise um den Graphen seiner Antwortfunktion partiell darzustellen.

Lässt man die $[0,1]$ -skalierten Entsprechungen der Niederschlagsmenge am Messpunkt „Schlitz“ in den letzten sieben Tagen (X_1) und die mittlere Lufttemperatur in den 4 Wochen (X_5) vor der Prognose so in den durch die Lerndaten gegebenen Wertebereichen variieren, dass ein Gitter von 400 gleichmäßig im Intervall $[0,1]^2$ verteilten Eingabepunkten entsteht und werden die restlichen fünf Netzeingabegrößen konstant auf den Werten

des Testdatensatzes mit der Nummer 472 (15.09.1999) gehalten, lässt sich der Graph dieser (im weiteren Sinne) *partiellen Netzausgabefunktion* wie in Abb. 4.36 darstellen.

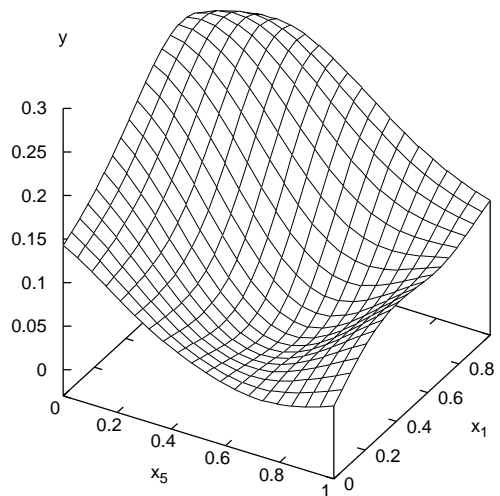


Abbildung 4.36: Darstellung des dreidimensionalen Graphen der Ausgabefunktion des RBF-SOM-Netzes mit veränderlicher Niederschlagsmenge an der Mess-Station „Schlitz“ in den letzten sieben Tagen (mit $[0,1]$ -skalierten Werten x_1) und der mittleren Lufttemperatur in den vier Wochen ($[0,1]$ -skalierte Werte x_5) vor dem Prognosezeitpunkt; die restlichen fünf Eingabegrößen wurden konstant auf den Werten des Testdatensatzes 472 (15.09.1999) gehalten.

Die Grafik zeigt, dass sich der niedrige Breitenbach-Abfluss bei hohen Lufttemperaturen im Spätsommer im Tagesmittel auch bei ergiebigen Regenfällen kaum ändern würde. Bei kühler Witterung über die vier Wochen vor Mitte September hingegen — man braucht dafür nicht den als unwahrscheinlich einzuschätzenden Bereich von unter $x_2 \leq 0.2$ zu betrachten, der für diese Jahreszeiten zu niedrigen Temperaturen bis hin zu Minusgraden entspräche — können starke Regenfälle hingegen den zu erwartenden Abfluss stark beeinflussen, sodass mehr als ein Viertel des maximalen Abflusses erreicht werden kann. Diese Grafik beinhaltet keine Angaben über die für die einzelnen Prognosen zu erwartende Zuverlässigkeit. Man kann aber annehmen, dass die beiden genannten Eingabegrößen unter natürlichen Bedingungen nicht einmal näherungsweise ihre Extremwerte annehmen könnten, wenn die anderen konstant gehalten würden, da nicht alle Eingabegrößen voneinander unabhängig sind.

Die zum Teil starke Abhängigkeit der Prädiktoren voneinander und die Unmöglichkeit gewisser Kombinationen von Eingabewerten verdeutlicht Abb. 4.37. Das Ergebnis einer Korrelationsanalyse hätte eine geringere Aussagekraft als die linke Abbildung. Vor allem die Kombinationen von Werten x_1 und x_2 mit $x_1 > x_2$ sind (bis auf Rundungsfehler) nicht beobachtet worden. Dieser Sachverhalt hat eine einfache Erklärung: Es kann in sieben Tagen nicht mehr Niederschlag gefallen sein als in den sie überdeckenden zwei Wochen. Das bedeutet, dass jeder Wert der partiellen Netzausgabefunktion, der auf Eingabewertekombinationen mit der Eigenschaft $x_1 > x_2$ beruht, als unzu(ver)lässig eingestuft werden

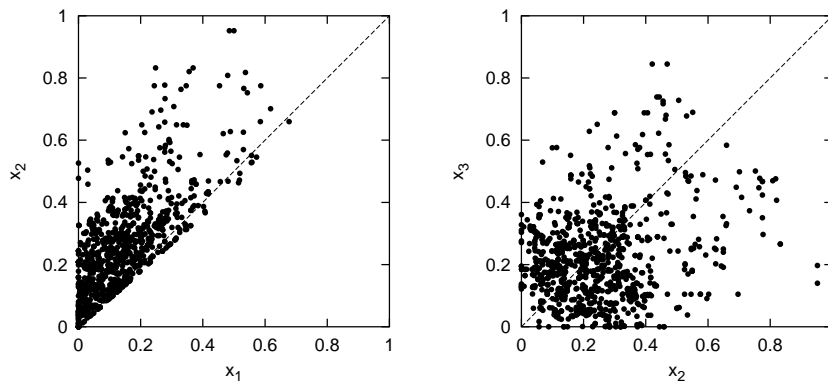


Abbildung 4.37: Streudiagramme der Testdaten für die Netzeingabegrößen X_1 und X_2 (links) sowie X_2 und X_3 (rechts).

muss. Würden die Werte beider Größen auf der Grundfläche in einer zu Abb. 4.36 ähnlichen Grafik liegen, wäre ein großer Teil der Ausgabefunktion nicht durch Datenpunkte gestützt. Hingegen besteht kaum eine Abhängigkeit der Größen X_2 und X_3 , die den in Schlitz gemessenen Niederschlag der letzten 14 Tage vor einer Prognose und der zwei Wochen davor beschreiben, wie die rechte Abbildung verdeutlicht.

Gemeinsam mit der partiellen Netzausgabefunktion in Abhängigkeit von nur einer Veränderlichen und 6 konstant gehaltenen Größen lassen sich die von *Visualrbfn* ausgegebenen Maße der Zuverlässigkeit darstellen. Ein Beispiel ist in Abb. 4.38 zu sehen.

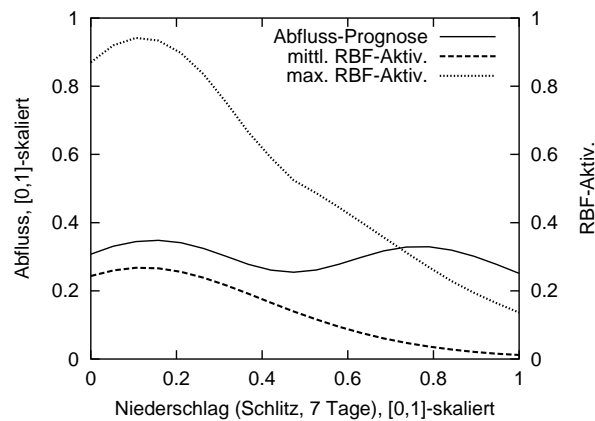


Abbildung 4.38: Partielle Ausgabefunktion des RBF-SOM-Netzes mit veränderlicher Niederschlagsmenge am Messpunkt „Schlitz“ in den sieben Tagen vor der Prognose; die restlichen sechs Eingabegrößen wurden konstant auf den Werten des Testdatensatzes 272 (27.02.1999) gehalten. Zusätzlich sind die maximale und die mittlere Aktivität aller RBF-Neuronen zu sehen.

Wenn in der Woche vor dem 27.02.1999 wesentlich mehr oder weniger — im Rahmen der beobachteten Werte — Niederschlag gefallen wäre, hätte sich der Abfluss kaum geändert. Dieser Sachverhalt lässt sich auch generell formulieren. Der Einfluss des Niederschlages

auf den erwarteten Abfluss des Breitenbaches unter ähnlichen wie den Ende Februar 1999 herrschenden Bedingungen ist gering. Gleichzeitig ist die Gültigkeit der Aussage aufgrund der abgeschwächten Aktivitäten der RBF-Neuronen umso stärker in Frage gestellt, je weiter man sich von den unter natürlichen Umständen zu erwartenden Bedingungen entfernt. Auch hier sind Werte $x_1 > 0.2325$ unter den oben genannten Voraussetzungen eigentlich nicht möglich, vgl. die linke Grafik in Abb. 4.37.

Der Knick in der Kurve der maximalen RBF-Aktivität in der Nähe von $x_1 = 0.5$ erklärt sich dadurch, dass unterschiedliche RBF-Neuronen ihre maximale Aktivität lieferten.

Die Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sowie die Anzahl der in die betreffenden Einzugsgebiete der SOM-Sieger fallenden Lerndaten ($\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$), die stückweise konstant und unstetig ist, liefern in Abb. 4.39 ähnliche Aussagen. Allerdings

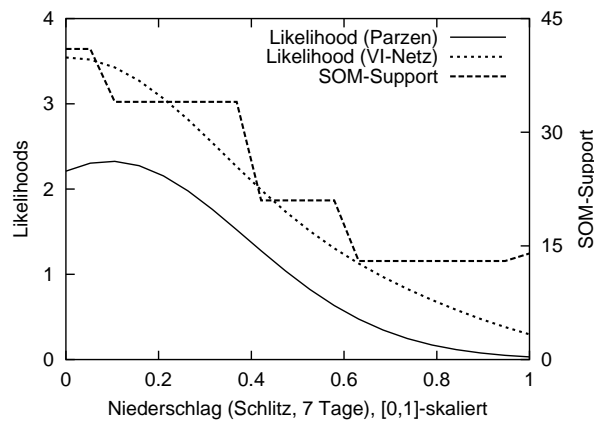


Abbildung 4.39: Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sowie die hier als Support bezeichnete Größe $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$, die stückweise konstant und unstetig ist. Ihre Werte wurden hier an 20 Stützstellen ermittelt und in dieser Grafik durch einen Polygonzug verbunden; vgl. Abb. 4.38.

würde die letztgenannte Größe für eine weitere Extrapolation nicht wie die anderen beiden weiter abnehmen und somit diese Eingaben nicht als neuartig anzeigen.

Die durch $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ gegebenen Prognoseintervalle sowie die erwarteten Fehlerbalken nach dem Predictive-Error-Bars-Ansatz, vgl. S. 20, sind gemeinsam mit der Netzwerkprognose in Abb. 4.40 zu sehen.

Die folgenden Abbildungen 4.41 bis 4.45 betreffen das Szenario, das der Frage nachgeht, welche Abflussprognosen vom RBF-SOM-Netzwerk und welche Werte der Zuverlässigkeitsmaße ausgegeben würden, wenn ausgehend von ähnlichen Bedingungen, wie sie beispielsweise am 27. Februar 1999 (Testdatensatzes 272) herrschten, sich die Lufttemperatur über vier Wochen vor der Prognose an „Haus B“ am Breitenbach geändert hätte.

Die Abflussprognosewerte fallen für bis zu etwa -9 °C entsprechende Eingabewerte genauso wie für sehr hohe Temperaturen (max. über 20 °C) ab, wie aus Abb. 4.41 ersichtlich ist. Allerdings gilt dies auch für die maximale Aktivität der RBF-Neuronen. Unter diesen

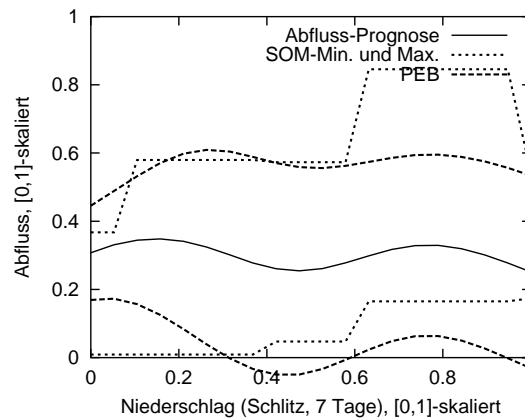


Abbildung 4.40: Abfluss-Prognose in Abhängigkeit von Werten x_1 und sonst konstanten Größen, erwartete Fehler nach dem Fehlerbalken-Ansatz (PEB) sowie $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ als Minima und Maxima der SOM-Klassen; vgl. Abb. 4.38.

Bedingungen würde erfahrungsgemäß auch der Niederschlag nicht konstant sein und er wäre bei sehr tiefen Temperaturen nicht unmittelbar für den Abfluss wirksam.

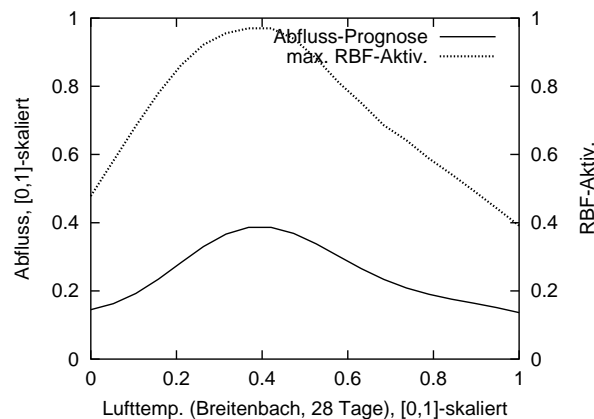


Abbildung 4.41: Partielle Ausgabefunktion des RBFSOM-Netzes und maximale Aktivität der RBF-Neuronen in der verdeckten Schicht des RBFSOM-Netzes mit veränderlicher Lufttemperatur, die in den vier Wochen vor der Prognose an „Haus B“ am Breitenbach gemessen wurde; die restlichen sechs Eingabegrößen wurden konstant auf den Werten des Testdatensatzes 272 (27.02.1999) gehalten.

Auch in Abb. 4.42 verlaufen die Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sehr ähnlich. Die unstetige Funktion $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$ zeigt jedoch ein geradezu gegenläufiges Verhalten.

Die Abfluss-Prognose zusammen mit den erwarteten Fehlern nach dem Fehlerbalken-Ansatz sowie die aus der SOM resultierenden Größen $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ sind in Abb. 4.43 gezeigt.

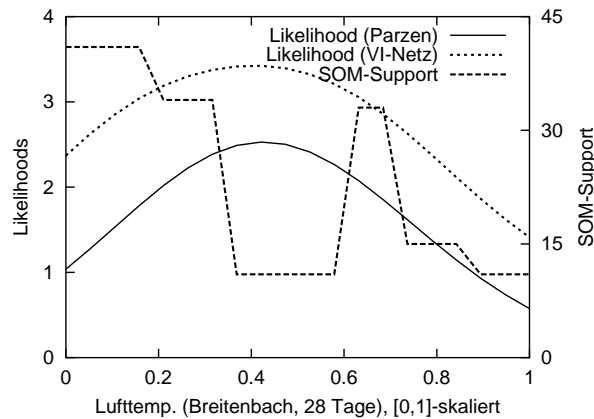


Abbildung 4.42: Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sowie die hier als Support bezeichnete Größe $\#\mathcal{X}_{\kappa(x(t))}^L$. Deren Werte, die an 20 Stützstellen berechnet und durch einen Polygonzug verbunden wurden, sind stückweise konstant und unstetig; vgl. Abb. 4.41.

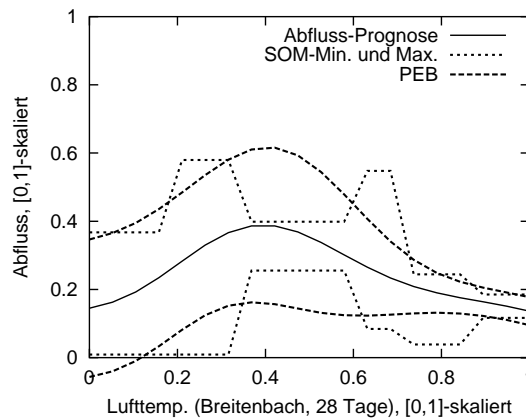


Abbildung 4.43: Abfluss-Prognose, erwartete Fehler nach dem Fehlerbalken-Ansatz (PEB) sowie $\min(\mathcal{Y}_{\kappa(x(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(x(t))}^L)$; vgl. Abb. 4.41.

Die in Abb. 4.44 dargestellten Schätzungen der Konfidenz- und Prognoseintervalle durch das VI-Netz ändern sich im betrachteten Intervall nur geringfügig. Das 95%-Konfidenzintervall ist sehr klein. Ein wesentlicher Grund dafür liegt hier in den großen Werten der Wurzeln der im Sinne der RBFs wirksamen Anzahl von Datenpunkten in Gl. 2.10 (S. 21). Da die Lage des bedingten Mittelwertes aber nicht derart abgesichert ist, muss dieses Ergebnis kritisch betrachtet werden.

Die Sieger der SOM für dieses Szenario waren die SOM-Zellen mit den Indizes 3, 4, 17, 30, 43, 56. Die Trajektorie der Siegerneuronen ist in Abb. 4.45 zu sehen. Die Darstellung lässt sich mit den Eingabevektor-Komponenten in Abb. 4.10 (S. 86, Grafik für x_5) und 4.11 (S. 86, dabei in jeder Linien-Grafik der fünfte Punkt von links) vergleichen. Vor allem die Werte der korrelierten Größe X_6 würden sich im Szenario normalerweise ebenfalls verändern, wie die Abbildungen zeigen.

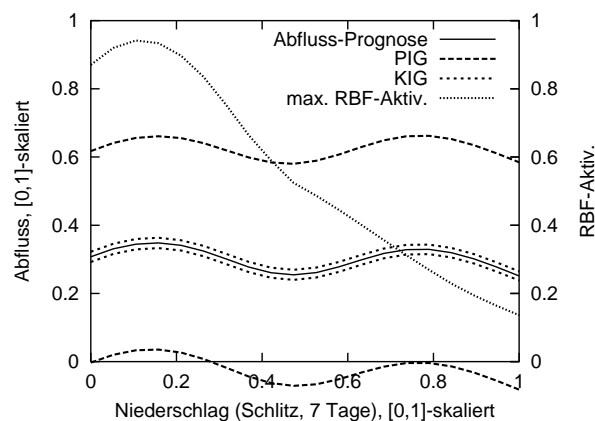


Abbildung 4.44: Ausgaben des VI-Netzes mit Schätzungen der Konfidenz- und Prognoseintervalle gemeinsam mit der Abflussprognose und zusätzlich der maximalen Aktivität der RBF-Neuronen, vgl. Abb. 4.41. Die Abkürzungen PIG und KIG stehen für die Prognose- bzw. Konfidenzintervallgrenzen.

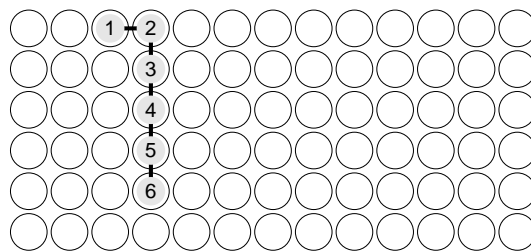


Abbildung 4.45: Trajektorie der SOM-Sieger in der verdeckten Schicht des RBFSOM-Netzes für das Szenario mit sich im gemessenen Wertebereich verändernden, über vier Wochen gemittelten Lufttemperaturen am Breitenbach und sonst konstanten Netzeingaben.

Die Wichtigkeit der Prädiktoren ließ sich über den RMSE einschätzen, indem während der Prognosen für die Testdaten jeweils eine Netzeingabegröße auf ihrem Mittelwert gehalten wurde. Je wichtiger sie war, desto größer musste der Fehler werden. War sie umgekehrt irrelevant (oder auch nur redundant), veränderte sich der Fehler kaum. Im Einzelfall konnte sich die Generalisierungsfähigkeit des Netzes sogar verbessern. Wie Abb. 4.46 zeigt, ist die Größe X_6 , die für die Lufttemperatur über die letzten 80 Tage am Breitenbach steht, verhältnismäßig wichtig, während die Lufttemperatur der letzten vier Wochen vor dem Prognosedatum — vgl. Abb. 4.38 — eine vergleichsweise geringe Relevanz aufweist.

Durch die automatische Reskalierung der Prädiktoren mit Hilfe des in Unterabschnitt 2.3.3 (S. 30) beschriebenen Gradientenabstiegsverfahrens auf der Basis der Fehlerrückführung konnte die Generalisierungsfähigkeit des Modells verbessert werden. Die Anzahl der Neuronen in der verdeckten Schicht des RBFNs wurde auf 28 verringert, da das Verfahren mit der in den oben dokumentierten Experimenten verwendeten Anzahl von RBF-Zentren nicht konvergierte. Um die Laufzeit kurz zu halten, wurden die zum Trai-

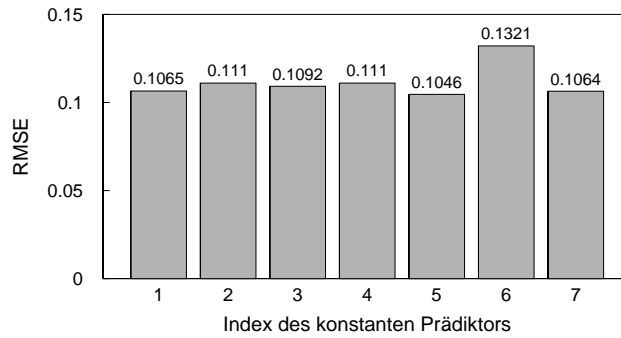


Abbildung 4.46: Abschätzung der Wichtigkeit der Prädiktoren für das auf einem RBFSOM-Netz basierende Modell, indem in den Testdaten jeweils einer auf seinen Mittelwert gesetzt und der RMSE mit den auf diese Weise modifizierten Daten berechnet wurde. Der RMSE mit allen sieben Prädiktoren betrug $E_{\text{RMS}} = 0.1057$.

ning verwendeten Daten durch die Auswahl jedes vierten Lerndatensatzes reduziert. Das Netz wurde als RBFSOM mit festen Bandbreiten von $h = 0.3$ hybrid und anschließend durch das Gradientenabstiegsverfahren überwacht trainiert. Die Lernraten η_s , η_w und η_z betragen 0.001, der Momentum-Faktor μ war gleich 0.8 gesetzt. Nach 170 Iterationen konnte der Lernfehler auf einen RMSE von 0.0685 verringert werden. Der Testfehler lag anschließend mit dieser Konfiguration bei $E_{\text{RMS}} = 0.0999$ ($B = 0.4138$). Die Skalierungsfaktoren hatten für die in Tab. 4.2 (S. 83) genannten Prädiktoren die Werte 1.02, 1.05, 1.08, 0.96, 1.05, 1.05 und 1.12 angenommen. Damit wurde erneut ein RBFSOM-Netz mit für jede RBF durch eine 2-Nächste-Nachbarn-Heuristik bestimmten Bandbreiten und allen Lerndaten trainiert. Der Test-RMSE betrug schließlich $E_{\text{RMS}} = 0.0887$ ($B = 0.5373$).

Für die Prognosen des Breitenbachabfluss-Tagesmittels mit einer Auswahl von 18 der oben genannten Prädiktoren sowie dem Abfluss des Vortages und der Abflussänderung zum davor liegenden Tag diente ein RBFSOM-Netzwerk mit 100 RBF-Zentren und weiteren linearen Neuronen, vgl. Gl. 2.5 (S. 18). Zum Training der RBF-Zentren wurde eine 10×10 -SOM benutzt.

Im Vergleich zu Abb. 4.19 zeigt Abb. 4.47 eine wesentlich größere Übereinstimmung zwischen den Messwerten und den Prognosen, sodass man beide in der Zeitreihen-Darstellung (links) teilweise kaum unterscheiden kann. Für die Prognosen im Bereich der Testdatensätze mit den Nummern zwischen 908 und 918 lagen keine Abflusswerte vor. Der RMSE von 0.0358 bedeutet, dass man bei der Vorhersage des täglichen mittleren Breitenbachabfluss-Tagesmittels einen mittleren Fehler von $0.0358 \cdot 119 \text{ l/s} \approx 4.26 \text{ l/s}$ erwarten kann. Das Bestimmtheitsmaß war mit $B = 0.9245$ recht hoch.

Allerdings erzielte die naive Prognose $\check{y}(t) = y(t-1)$ für die Testdaten ein Ergebnis von $E_{\text{RMS}} = 0.0341$ und war damit im Mittel besser als das Neuronale Netz, was sich auch in dem als *Theilkoeffizient* bezeichneten Verhältnis der Fehlerquadratsummen der

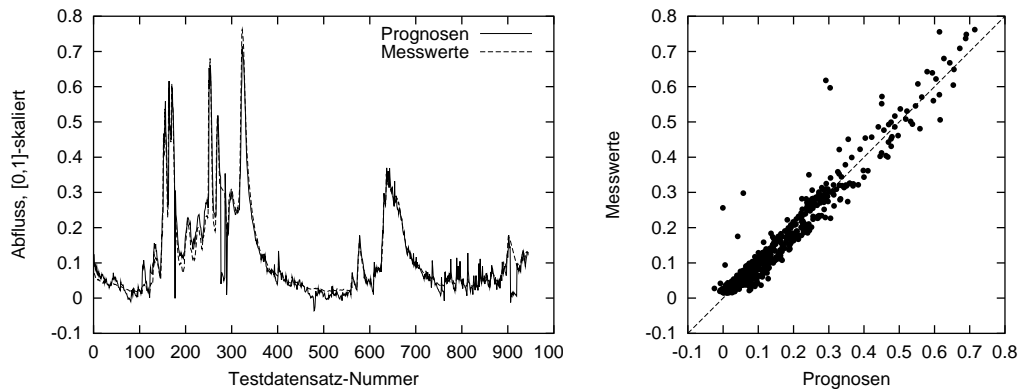


Abbildung 4.47: Messwerte und Prognosen des RBFSOM-Netzwerkes mit zusätzlichen linearen Zellen des mittleren täglichen Breitenbachabflusses auf der Basis von 20 Prädiktoren, darunter die Vortageswerte des Abflusses und ihren Änderungen zum davor liegenden Wert. Eine Einheit auf der Ordinatenachse entspricht in der ursprünglichen Skala 119 ℓ/s ; die Testdatensatz-Nummern zählen die Tage ab dem 01.06.1998; die Abszisse des linken Diagramms kann daher als Zeitachse aufgefasst werden.

RBFSOM-Prognosen gegenüber den Vorhersagen des einfachen Modells von 1.0499 widerspiegelte. Das Bestimmtheitsmaß für die naive Prognose betrug $B = 0.9318$. Auch zahlreiche, mit der Software *Trajan* simulierte LNNs, GRNNs, RBFNs und MLPs lieferten keine besseren Modelle. Die Prognose der täglichen Abflussänderung, bei der die Differenz $\Delta_y(t) := y(t) - y(t - 1)$ als abhängige Größe durch die Netzausgabe vorhergesagt wurde, brachte ebenfalls keine wesentliche Verbesserung der globalen Vorhersagegüte. Der Theilkoeffizient betrug nach der Addition der Änderungen und des Vortages-Abflusswertes für die Testdaten 1.0334.

4.2.5 Diskussion

In der vorliegenden Fallstudie konnten viele der Möglichkeiten, aber auch einige Schwierigkeiten bei der Anwendung des RBFSOM-Netzes mit der Software *Visualrbfn* illustriert werden. An dieser Stelle werden die Aspekte diskutiert, welche speziell die Problematik bei Abflussprognosen betreffen. In Kapitel 5 (S. 111 ff.) setzt sich die Diskussion unter allgemeineren Gesichtspunkten der Anwendung des RBFSOM-Netzes fort.

Das für die meisten Leser aus der Anschauung nachvollziehbare Wetter-Abfluss-Geschehen von Fließgewässern, hier am Beispiel des Breitenbaches, ist von grundsätzlicher Bedeutung. Prognosen über die Hochwassergefahren von Flüssen und Strömen lassen sich verbessern, indem Vorhersagen über die in sie mündenden Zuflüsse abgegeben werden. Hochwasserwarnungen für das Forschungsobjekt Breitenbach können einen Beitrag zum Schutz wertvoller Messeinrichtungen und aufwendiger Freilandexperiment-Aufbauten leisten. Auf die Bedeutung des Abflusses als Umweltfaktor unter ökologischen Gesichtspunkten wurde bereits in Abschnitt 4.1 und in der Einleitung zu dieser Fallstudie

hingewiesen.

Mit Hilfe des vorliegenden Wetter-Abfluss-Modells können Lücken in den Datensätzen geschlossen werden. Ausreißer im Sinne fehlerhafter Messungen lassen sich sowohl in der Lern- als auch in der Arbeitsphase des Netzes erkennen und — wie in der Vorverarbeitung zu dieser Fallstudie geschehen — eliminieren oder durch die Netzausgabe ersetzen. Aber auch für das KNN-Modell außergewöhnliche und für die im Biotop Breitenbach lebenden Organismen unerwartete, wirklich aufgetretene, extreme oder auch nur bezüglich des Zeitpunktes unwahrscheinliche Ereignisse sind als „Störungen“ gegenüber dem zu erwartenden Abflussverhalten des Baches, an die sich sowohl das KNN als auch die *Biozönose* mehr oder weniger gut angepasst haben, zu erkennen. Größere Zeitabschnitte mit fehlenden Messwerten in den Datensätzen lassen sich mit den aufgrund vorliegender Abflusswerte genaueren Prognosen füllen, indem die Netzausgabe eines Tages als Netzeingabe für die Vorhersage des nächsten Tages verwendet wird. Dabei müssten dann aber zumindest die Werte der anderen Prädiktoren als Messwerte oder in Form von z. B. Wettervorhersagen, vgl. [ZBS99], vorliegen. Wenn sich die Modellgüte wesentlich verbessern ließe, könnten die Netzwerkprognosen auf der Basis von Temperatur- und Niederschlagsdaten sogar die kontinuierliche Messung des Abflusses überflüssig machen.

Auf der Basis von Wetter-Abfluss-Modellen dieser Art sind Simulationen mit hypothetischen Eingabedaten möglich („Was wäre, wenn...?“), wobei die Zuverlässigkeit in die Prognosen als weitere Netzausgabe zur Verfügung steht. Denkbar sind u. a. Szenarien mit geringfügigen Klimaveränderungen, etwa um der Frage nachzugehen, wie sich das Abflussmuster des Breitenbaches durch eine Erwärmung mit höheren Durchschnittstemperaturen im Winter sowie heißeren oder auch feuchteren Sommern verändern würde.

Zugleich ist einzuräumen, dass die Güte täglicher Prognosen über den $2\frac{1}{2}$ Jahre dauernden Testzeitraum in der vorliegenden Fallstudie noch zu verbessern ist. Zwar folgten die Netzwerkprognosen auch ohne Informationen über vorangegangene Abflusswerte weitgehend — zeitweise aber bestenfalls in der Tendenz — dem tatsächlich beobachteten Verlauf des Abflusses und die Vorhersagen mit den als Netzeingaben dienenden Vortageswerten und ihrer Differenz zu dem davor liegenden Wert lieferten eine geringe Abweichung zu den gemessenen Werten. Trotzdem bedeutete im erstgenannten Fall ein Bestimmtheitsmaß von etwa 0.34 keine exzellente generelle Beschreibung des Zusammenhanges und für den anderen Fall bleibt zu klären, warum die naive Prognose durch den Abflusswert des Vortages eine im Mittel bessere Vorhersage als das aufwendig an die Lerndaten angepasste KNN lieferte. Als Konkurrenzmodell für ein KNN ist die naive Prognose nur dann sinnvoll, wenn die Information über den vorangegangenen Abflusswert auch in das KNN eingeht, sonst ist der Vergleich „unfair“.²¹⁾

Dabei ist der unterstellte Zusammenhang zwischen Temperatur- und Niederschlagsdaten sowie dem Breitenbachabfluss auf der Basis der vorliegenden Daten und der verwendeten Sampling-Rate mit Tageswerten generell schwierig zu beschreiben, wie aus der ähnlichen Güte der Modelle in zahlreichen Versuchen mit anderen Netzwerktypen (GRNN, MLP,

²¹⁾ Beispiele für „unfaire“, publizierte Vergleiche zwischen „traditionellen“ Methoden und KNNs sind in [Cha93] zu finden.

RBF, LNN) deutlich wurde. Ein Grund könnte sein, dass die naive Prognose im Mittel gute Vorhersagen liefert.²²⁾ Das heißt, dass sich der Abfluss des Breitenbaches von Tag zu Tag oftmals nur unwesentlich verändert. Wenn er starke Anstiege oder Abfälle zeigt, ist er aufgrund der gegebenen Prädiktoren im eigentlichen Sinne des Wortes kaum berechenbar. Bei nur wenige Stunden andauernden, heftigen Oberflächenwasserabläufen nach Starkregenereignissen (Wolkenbrüchen) werden in Abhängigkeit von der Uhrzeit und der zeitlichen Verzögerung beide am selben Tag oder an aufeinander folgenden Tagen stattfinden. Daher wurde den täglichen Niederschlagswerten keine hohe Relevanz für die Abflussvorhersage zugeordnet. Bei langfristigen Abflussveränderungen lässt sich hingegen nicht mit Bestimmtheit angeben, an welchem Tag die Zu- oder Abnahme in welcher Stärke eintritt. Hier könnte eine Veränderung der Sampling-Rate, etwa mit dem Ergebnis von Stundenwerten oder auch Wochenmitteln, Abhilfe schaffen.

Um bei radial-basierten Netzwerken die Anzahl von Prädiktoren gering zu halten und gleichzeitig ein ausreichend großes Zeitfenster zu betrachten, mussten in der Vorverarbeitung die Tageswerte der unabhängigen Größen zusammengefasst werden. Durch die Summenbildung gingen aber Informationen verloren. Über die Dauer, Stärke und Art von Niederschlagsereignissen lassen sich nach einer solchen Summenbildung kaum noch zuverlässige Aussagen treffen. Einzelne, zeitweise wichtige Einflussgrößen, z. B. auf die Verdunstungsraten wirkende Sonneneinstrahlung und Windverhältnisse, gingen nicht (direkt) in das Modell ein. Vor allem die Änderung des Abflusses des Breitenbaches bei und nach plötzlichen Schneeschmelzen ist kaum in guter Näherung vorhersagbar.

Die verwendeten Temperatursummen enthalten Informationen über den Vegetationsstatus und die mittlere *Evapotranspiration*.²³⁾ Durch die zusätzlich in dem Modell enthaltenen Werte der Temperaturen im aktuellen und dem vorhergehenden Mittel über jeweils fast ein viertel Jahr lässt sich die vorhandene schwache Saisonalität beschreiben. Diese ist vor allem durch einen geringen Abfluss im Sommer und im Frühherbst charakterisiert, auf den auch heftige Gewitter im Tagesmittel kaum Einfluss haben. Die Vegetation, der Waldboden und der darunter liegende Buntsandstein nehmen Feuchtigkeit auf, speichern sie und geben sie nur verzögert wieder ab. In der Folge wirkt sich nicht jedes Regenereignis unmittelbar auf den Abfluss aus. Im Gegensatz zu radial-basierten Netzen wäre bei Multi-Layer-Perzeptron-Netzen, im Spezialfall der LNNs mit linearen Neuronen, eine solche Summenbildung — unter der Voraussetzung eines ausreichend vorhanden Datenumfanges zum Anpassen der vielen Parameter — nicht nötig, da bei der Berechnung der Aktivität der Neuronen in den verdeckten Schichten bzw. in der Ausgabeschicht durch das Skalarprodukt gewichtete Teilsummen gebildet werden. Im Idealfall könnte der Zeitunterschied vom Regenereignis bis zum Wiederaustreten des Wassers aus der Quelle gefunden werden.

²²⁾ Der einfache Ansatz arbeitet nach dem Motto: „Es bleibt so, wie es ist.“ bzw. „Es wird morgen so, wie es heute war.“

²³⁾ Unter *Evapotranspiration* versteht man die Wasserdampfabgabe eines Pflanzenbestandes an die Atmosphäre, bestehend aus der nicht regulierbaren Verdunstung des Bodens (Evaporation) und niederer Pflanzen und der regulierbaren Transpiration höherer Pflanzen [Bro94, Bd. 6 (DS-EW), S. 684].

Trotz des Verwendens von Abflusswerten, die höchstens dem Über-das-Ufer-treten des Breitenbaches entsprachen,²⁴⁾ hatten extreme Abflussänderungen einen so starken Einfluss auf die Parameteranpassung ausgeübt, dass die Modelle in Zeiträumen mit eigentlich geringen Änderungsraten durch große Sprünge „übersensibel“ reagierten. Dieses Verhalten, das vor allem bei linearen Regressionsfunktionen auftritt, wird als „Hebeleffekt“ [Loh01] bezeichnet. Hier kann eine logarithmische Skalierung des Abflusses eine Verbesserung bringen.

Die verwendeten LNNs bzw. die linearen Approximationen der Regressionsfunktionen bildeten die besser generalisierende naive Prognose nicht nach, obwohl sie theoretisch dazu befähigt wären, indem der zu dem Prädiktor $y(t-1)$ gehörige Regressionskoeffizient gleich eins und alle anderen gleich null gewählt worden wären. Stattdessen passten sich die Modelle an die Trainingsdaten einschließlich der darin enthaltenen Extremwerte an und wiesen dann eine schlechtere Verallgemeinerungsfähigkeit auf.

Um die Vorteile der naiven Prognose und des RBFSOM-Netzes in einem Modell zu nutzen, kann untersucht werden, unter welchen Bedingungen das KNN bessere Prognosen als der simple Ansatz liefert. Im einfachsten Fall könnte daraus eine Regel resultieren, die darin besteht, dass man für Wetterverhältnisse mit Durchschnittstemperaturen über 18 °C und keinem Regen die naive Prognose verwendet. Die Entscheidung, welches Modell unter welchen Bedingungen verwendet werden soll, kann später ein anderes, klassifizierendes KNN treffen. Ein weiterer Ansatz ist die Erstellung eines Ensembles von Modellen, das ein (gewichtetes) Mittel der Prognosewertes des KNNs und des Abflusswertes des Vortages liefert.

Die an der etwa 5 km vom Breitenbach entfernt gelegenen Mess-Station in der Stadt Schlitz gemessene Niederschlagsmenge ist möglicherweise kein immer relevanter Prädiktor für das tatsächliche Geschehen im Einzugsgebiet des Breitenbaches, auch wenn die Korrelation (Maß-Korrelationskoeffizient nach PEARSON und BRAVAIS) zwischen den Niederschlagsmengen in Schlitz und bei Michelsrombach mit 0.9110 hoch war und der Breitenbach zwischen den beiden Mess-Stationen liegt.

Für die Berechnung der Generalisierungsfehler für die Testdaten spielten die wenigen fehlenden Messwerte eine untergeordnete Rolle.

Die in der vorliegenden Fallstudie gebildeten Modelle sind zunächst nur für den Breitenbach und sein Einzugsgebiet gültig. Für eine Übertragbarkeit auf andere Bäche dieser Größenordnung in der Nähe ist mindestens ein Neutraining des KNNs nötig. Für andere Fließgewässer müssen andere Prädiktoren ausgewählt und ein dafür optimiertes Netzwerk erzeugt und trainiert werden.

Untersuchungen an anderen Fließgewässern können kaum zum Vergleich der Ergebnisse und Methoden herangezogen werden. Die Abflüsse kleiner Bäche in Japan und den USA mit teilweise kleineren Einzugsgebieten als das des Breitenbaches wurden auf der

²⁴⁾ An der Mess-Station entspricht das etwa 120 ℓ/s . Extreme Hochwasserereignisse lagen bei mehr als dem fünffachen Wert. Am 23.01.1995 wurde ein Spitzenwert von über 600 ℓ/s und mehr als 386 ℓ/s im Tagesmittel geschätzt, wobei der Abfluss am Vortag noch 24 ℓ/s betrug und am dem Hochwasser folgenden Tag noch 145 ℓ/s entsprach.

Basis von Regenmengen der letzten Stunden vor der Prognose und bekannter Schüttungswerte ohne Berücksichtigung der Temperatur oder weiterer Einflussgrößen mit Hilfe von RBFNs und MLPs für die nächsten Stunden vorhergesagt [ZFH94, JF98]. In diesen Fällen kann nur ein fast unmittelbar auftretender Oberflächenabfluss beschrieben worden sein. Die Vorhersagen wöchentlicher Mittel von Fließgeschwindigkeiten, Abflusswerten und Pegelständen großer Flüsse erscheinen beispielsweise für den Winnipeg-River in Kanada [ZBS99] oder am unteren Mississippi in den USA [HZB02] wesentlich einfacher zu sein, vor allem, wenn es sich um Zeitreihen mit einer stark ausgeprägten Saisonalität und Gewässer mit vergleichsweise großen Einzugsgebieten handelt. Ein ebenfalls auf Tageswerten basierendes Regen-Abfluss-Modell mit MLPs lieferte in [LDDEG96] eine gute Prognosequalität. Allerdings ist das Regen-Abflussverhalten eines marokkanischen Wadis kaum mit dem des Breitenbaches zu vergleichen.

Das mehr oder weniger langsame Abfließen wurde mit dem bestehenden Modell schlecht abgebildet, wenn dem Modell Informationen über den Abfluss vorenthalten wurden. Daraus lässt sich ableiten, dass für solche Modelle ein autoregressiver Anteil oder eine Speichereinheit für vorangegangene Prognosen vorhanden sein sollte.

Die Vorhersagegenauigkeit für Zeitreihen der in der vorliegenden Fallstudie zugrunde liegenden Art könnte möglicherweise mit *partiell rekurrenten Netzwerken*, beispielsweise zur Modellierung dynamischer Systeme verwendeten ELMAN- und JORDAN-Netzen, vgl. [Fre94, S. 185 ff.] und [Zel94, S. 138 und S. 141] erhöht werden. Rekurrente SOMs in Kombination mit lokalen linearen Modellen existieren bereits zur Prognose von Zeitreihen [KVHK97]. Ein Ansatz zukünftiger Forschungen könnte daher darin bestehen, RBF-SOM-Netze durch eine der in den genannten Netzwerktypen vorhandenen *Kontextzellen* zu erweitern, sodass ein Gedächtnis der zuletzt ausgegebenen Werte des Netzes vorläge. Diese Kontextzelle würde daher ihre Eingabe von dem Ausgabeneuron erhalten und ihre Ausgaben entweder an die verdeckten, radialen Neuronen weiterleiten, was Probleme beim Training der RBF-Neuronen nach sich zöge, oder an das lineare Ausgabeneuron schicken. Für die Anwendung dieses Netzes dürften dann aber keine Lücken in den Eingabedaten vorhanden sein. Ob dieser Ansatz eine wesentliche Verbesserung gegenüber der Verwendung des vorigen Abflusswertes als Netzeingabe bringen würde, bliebe zu überprüfen.

5 Abschlussdiskussion

Einzelne Aspekte des RBF-SOM-Netzes, seiner Umsetzung in die Simulationssoftware *Visualrbfn* und der Anwendung in limno-ökologischen und hydrologischen Fallstudien wurden in dem Abschnitt 3.3 (S. 64) sowie in den Unterabschnitten 4.1.4 (S. 76) und 4.2.5 (S. 106) diskutiert. Im Folgenden werden weitere Ergebnisse und allgemeine Konzepte erörtert.

Der Einschätzung der Zuverlässigkeit von Prognosen durch ein auf der Basis von Messwerten (\mathbf{x}_i, y_i) für $i = 1, \dots, n$ gebildetes Regressionsmodell, das eine stochastische Abhängigkeit des reellwertigen Merkmals Y von einem Zufallsvektor \mathbf{X} beschreibt, kommt in vielen Anwendungen eine entscheidende Bedeutung zu.

Zur Beurteilung der *globalen* Prognosegüte und Generalisierungsfähigkeit lässt sich die Wurzel der mittleren Fehlerquadratsumme (RMSE) mit Testdaten, die im Gegensatz zu den Lerndaten nicht für die Anpassung des Regressionsmodells verwendet wurden, durch Hold-Out-, Kreuz- oder Bootstrap-Validierungen berechnen.

Auch wenn — wie im Falle von RBFNs — eine Skalierung der Ausgabegröße zum Anwenden des KNNs überflüssig erscheint, kann dadurch die Einschätzung des RMSEs vereinfacht werden. Zwar ist der mit den Größenordnungen der Variablen in der ursprünglichen Skala normalerweise vertraute Fachwissenschaftler beispielsweise daran interessiert zu erfahren, um wie viele Liter pro Sekunde die Prognose des Abflusses eines Fließgewässers erwartungsgemäß im Mittel von den tatsächlich zu messenden Werten abweicht oder auf wie viele hundert Insekten genau man die Abundanz in einer Emergenzfall vorhersagen kann. Allerdings lässt sich die Vorhersagequalität im Allgemeinen besser beurteilen, wenn der RMSE für [0,1]-skalierte Größen angibt, um wie viel Prozent der Variationsbreite sich die Netzwerkausgaben im Mittel verschätzen oder wenn der RMSE für standardisierte Variable darauf hinweist, welchen Anteil der Streuung man durch das Modell erklären kann. Mit gegebenen Dispersionsmaßen in der ursprünglichen Skala lassen sich die Angaben zumeist leicht umrechnen.

Einige Gütemaße basieren auf einem Vergleich der Prognosen $\hat{y}_i := \hat{y}(\mathbf{x}_i)$ zu denen $\check{y}_i := \check{y}(\mathbf{x}_i)$ von einfacheren Konkurrenzmodellen, die einen Mindestanspruch für die Qualität der Prognosen vorgeben, indem die Fehler auf Testdaten (\mathbf{x}_i, y_i) beider ins Verhältnis gesetzt werden:

$$E_{\text{rel}} = \frac{\sqrt{\sum_{i=1}^n (y_i - \hat{y}_i)^2}}{\sqrt{\sum_{i=1}^n (y_i - \check{y}_i)^2}}. \quad (5.1)$$

Die Prognosen \hat{y}_i sind im Mittel besser als die Werte \check{y}_i , wenn E_{rel} kleiner als eins ist. Die einfachste Wahl für \check{y} ist der Mittelwert \bar{y} . Sie ist in Gl. 4.1 (S. 72) dargestellt und als $E_{\text{rel}}^{\bar{y}}$ bezeichnet. Für standardisierte (z-transformierte) Größen kommt $E_{\text{rel}}^{\bar{y}}$ dem RMSE nahe, da die Standardabweichung, die nach der Erweiterung des Bruches um $\sqrt{1/n}$ im Nenner steht, auch für die Testdaten in der Nähe von eins liegen sollte. Das prinzipiell sehr ähnlich aufgebaute Bestimmtheitsmaß ist aus $E_{\text{rel}}^{\bar{y}}$ über die Beziehung $B = 1 - E_{\text{rel}}^{\bar{y}^2}$ zu berechnen.

Für die Prognose von Zeitreihen ist E_{rel} ein Theilkoeffizient, wenn für die nur jeweils den nächsten Zeitschritt vorhersagende *Einschrittprognose*^(e) des Testdatensatzes i die naive Prognose $\check{y}_i = y_{i-1}$ verwendet wird, vgl. Unterabschnitt 4.2.4 auf S. 105. Als Konkurrenzmodell ist auch *exponentielles Glätten*^(e)

$$\check{y}_i = \acute{y}_{i-1} = \alpha y_{i-1} + (1 - \alpha) \acute{y}_{i-2}, \quad (5.2)$$

modifiziert nach [Voß00, S. 264], mit einem günstig zu wählenden $\alpha \in [0, 1]$ denkbar.¹⁾ Für den Fall des Gedächtnisverlustes durch $\alpha = 1$ schließt dies die naive Prognose ein. Für $\alpha = 0$ wird immer wieder der Startwert prognostiziert, der nach [Voß00, S. 265] beispielsweise als arithmetisches Mittel gewählt werden kann. Damit wäre der erwähnte Fall $\check{y}(\mathbf{x}_i) = \bar{y}$ beschrieben. Bei Zeitreihen mit einer starken *saisonalen Komponente* kann — wenn nicht in der Vorverarbeitung der Daten eine *Saisonbereinigung* durchgeführt wurde — der Vergleich mit dem *Phasendurchschnitt* sinnvoll sein [OWWS01, S. 210].

Zur vergleichenden Beurteilung der Prognosen durch unterschiedliche KNNs kann E_{rel} ebenfalls herangezogen werden. Wegen der einfachen und schnellen Erzeugung sollte man nach der Bildung aufwendiger Modelle mit Prognosen $\hat{y}(\mathbf{x})$ möglichst auch die Ergebnisse $\check{y}(\mathbf{x})$ der linearen Approximation einsetzen, die sich z. B. durch LNNs ergeben. Eine ebenfalls einfach anzulegende Messlatte für Regressionsmodelle stellen GRNNs dar. Die im Rahmen der vorliegenden Arbeit entwickelte Software *Visualrbfn* liefert neben vielen anderen Informationen auch ein Regressionsmodell, das sich durch die Mittelwerte $\bar{y}_{\kappa(\mathbf{x})}$ der Elemente in jeder der m durch eine SOM gebildeten Klassen \mathcal{Y}_k^L ergibt. Es ähnelt *motorischen Karten*, wie sie in [RMS90, S. 115] beschrieben sind, sowie *Regressogrammen* mit festen Klassengrenzen, vgl. [Mic92, S. 19], [Pla93] und [Oba98, S. 79]. Im Unterschied zu den Letzteren werden aber keine Hyperquader, sondern Voronoi-Polygone zur Einteilung des Raumes der unabhängigen Größen verwendet.

Eine wesentliche Eigenschaft der beschriebenen Klassenmittel $\bar{y}_{\kappa(\mathbf{x})}$ und von GRNNs besteht darin, dass ihre Prognosen als ungewichtete oder gewichtete Mittelwerte im Wertebereich der Lerndaten y_i liegen. Bei Abflussvorhersagen eines Baches, wie in Abschnitt 4.2 beschrieben, wären beispielsweise keine negativen Abflusswerte möglich. Andererseits würde auch bei beliebig ergiebigen Wolkenbrüchen der maximal in den Trainingsdaten gemessene Abflusswert nie überschritten. Vor allem bei der Extrapolation unterscheiden sie sich damit von RBFNs, MLPs (vor allem mit nicht-sigmoiden Neuronen in der Ausgabeschicht) und dem Spezialfall der LNNs. Welches Verhalten ein Vorteil

¹⁾ Die Einschränkung $0 < \alpha < 1$ aus [Voß00, S. 264] wird hier erweitert.

oder Nachteil ist, entscheidet die Anwendungssituation. Bei KNNs, die beispielsweise aufgrund linearer Ausgabeneuronen Werte außerhalb des Lerndaten-Wertebereiches der abhängigen Größe vorhersagen, können ggf. im Rahmen einer Datennachbearbeitung aufgrund der Kenntnisse über das zu modellierende System oder über die Wertebereiche unmöglich auftretende Werte verworfen werden. Beispielsweise lassen sich bei den in Kapitel 4 beschriebenen Abundanz- und Abflussprognosen bezüglich der ursprünglichen Skala negative Werte auf den Mindestwert oder auf null setzen.

Bei der Verwendung der vorgestellten RBFSOM-Netze sind globale Gütemaße nicht nur für das überwachte Lernen der abhängigen Größe Y anzugeben, sondern auch zur Beurteilung der Repräsentationsfähigkeit von SOMs für Netzeingaben. Ein Beispiel ist auf S. 15 durch den *Quantisierungsfehler* gegeben. Überdies lässt sich insbesondere für große SOMs das *mittlere Distorsionsmaß* [KHKL96, S. 7] anwenden. Weitere Maße für die globale Zuverlässigkeit von SOMs bezüglich der Stabilität der Quantisierungsfehler und der Nachbarschaftsbeziehungen mit Hilfe von Bootstrap-Verfahren finden sich in [CBV01], wobei auch Verfahren zur Optimierung der Anzahl von SOM-Neuronen genannt werden.

Neben den genannten Maßen zur Einschätzung der *mittleren, globalen* oder *Gesamt-Qualität* von Regressionsmodellen wurden im Rahmen dieser Arbeit insbesondere Ansätze zur Beurteilung der *lokalen* oder *punktweisen Zuverlässigkeit* hybrider Prognose-systeme untersucht. Dabei haben für eine Prognose $\hat{y}(\mathbf{x}_0)$ vor allem zwei, weitgehend voneinander unabhängige Komponenten eine wesentliche Bedeutung. Erstens stellt sich die Frage, wie gut ein tatsächlich zu beobachtender Wert y_0 durch $\hat{y}(\mathbf{x}_0)$ beschrieben wird und damit, wie groß der zu erwartende Fehler an der Stelle \mathbf{x}_0 ist. Dabei spielt die lokale Streuung der abhängigen Größe ebenso eine Rolle, wie die Nähe des geschätzten Wertes zu dem zu schätzenden bedingten Erwartungswert. Zweitens besteht bei neuartigen, dem Prognosesystem unbekanntem Vektoren \mathbf{x}_0 die Gefahr, dass \mathbf{x}_0 fehlerhaft ist und außerhalb des Gültigkeitsbereiches des Modells liegt und weiterhin, dass die Prognose $\hat{y}(\mathbf{x}_0)$ bei einer Extrapolation unzuverlässig ist, da das Modell dafür nicht oder bei nur wenigen ähnlichen Lernmustern nur unzureichend angepasst wurde.

Die Hybridisierung von unüberwacht lernenden SOMs und überwacht trainierten RBFNs, wobei die SOM-Kodebuchvektoren \mathbf{z}_k ($k = 1, \dots, m$) als RBF-Zentren verwendet werden und die \mathbf{z}_k über die Nächster-Nachbar-Klassifizierung (bei gleichem Abstand entscheidet der kleinere Index) den Eingaberaum in m Klassen partitionieren, stellen einen Lösungsansatz für die Problemstellung dar. Die Anzahl der in einer Klasse $\mathcal{X}_{\kappa(\mathbf{x})}^L$ liegenden Lerndaten liefert einen ersten Hinweis darauf, durch wie viele ähnliche Lerndaten eine Prognose $\hat{y}(\mathbf{x})$ gestützt wird. Für jede Klasse $\mathcal{Y}_{\kappa(\mathbf{x})}^L$ lassen sich die dadurch bedingten Verteilungen sowohl durch Box-Whisker-Plots als auch durch weitere Kennwerte wie Lageparameter (z. B. der Mittelwert $\bar{y}_{\kappa(\mathbf{x})}$) und Dispersionsmaße beschreiben. Die Berechnung dieser Werte ist einfach und schnell durchzuführen. Eine große Ausdehnung der Klassen \mathcal{X}_k^L , die teilweise sogar unbeschränkt sind, schwächt die Aussagekraft für die Abschätzungen der Unterstützung der Prognose durch ähnliche Lerndaten und für die Berechnung lokaler Streuung der Ausgabegröße allerdings ab. Weiterhin können die durch die SOM gebildeten Klassen u. a. bei dem Vorliegen weniger Lerndaten, bei der Verwen-

dung einer großen SOM oder bei schlecht trainierten Netzen sehr klein sein, sodass nur wenige Datenpunkte in die Klassen fallen. Bei der Abschätzung des „Supports“ durch die Anzahl von Lern-Eingabedaten, die in den jeweiligen Klassen liegen, kann die Form und die Ausdehnung des Einzugsgebietes als Voronoi-Polyeder stark von der Hyperkugel abweichen, in der das zugehörige RBF-Neuron eine deutliche Aktivität zeigt. Bei zu kleinen Bandbreiten oder großen Einzugsgebieten der SOM-Neuronen kann dieser „Support“ überschätzt werden, während sehr große Bandbreiten oder sehr kleine Einzugsgebiete eine zu niedrige Punktedichte anzeigen. Daher sollten für jede RBF unterschiedlich wählbare Bandbreiten zumindest über eine Nächste-Nachbarn-Heuristik an die mittlere Ausdehnung der SOM-Klassen angepasst sein.

Die durch das RBFSOM-Netzwerk vermittelte Funktion \hat{y} ist mit als stetig vorausgesetzten Basisfunktionen im Gegensatz zu den erwähnten Mittelwerten der Klassen \mathcal{Y}_k^L sowie den Ausgaben von Counterpropagation-Netzwerken und den mit SOMs gekoppelten linearen Assoziatoren stetig, vgl. [Roj93, S. 372] und [NKK94, S. 113].²⁾ Diese günstige Eigenschaft teilen aber die bisher genannten SOM-basierten Größen nicht, deren Werte für jedes Einzugsgebiet eines SOM-Neurons konstant sind. Um zusätzlich stetige Funktionen zur Einschätzung der Dichte zu verwenden, wurde das RBFSOM-Netzwerk mit einem nicht-parametrischen Parzen-Dichteschätzer, dem VI-Netz, das ebenfalls die Dichten und überdies Prognose- und Konfidenzintervalle schätzt, und einem Verfahren zur Erzeugung lokaler Fehlerbalken verknüpft. Der Quantisierungsfehler stellt für beliebige Netzeingaben ein einfaches Extrapolationsmaß dar. Da die Werte der RBFs an einer Stelle x von dem Abstand zu den RBF-Zentren bzw. SOM-Kodebuchvektoren abhängen, liefern die maximalen Aktivitäten von RBF-Neuronen mit lokalen Basisfunktionen und konstanten Bandbreiten zu den Quantisierungsfehlern redundante, für unterschiedliche Bandbreiten zumindest ähnliche Informationen.

Mit *Visualrbfn* können die Zentren der RBFs durch verschiedene Verfahren positioniert werden. Bei einer zufälligen Auswahl der Zentren aus dem Eingaberaum oder selbst aus dem kleinsten Hyperquader, der die Daten beinhaltet, ist nicht auszuschließen, dass viele RBFs weit außerhalb der Punktwolke liegen und daher schlecht zu trainieren sind. Die zufällige Selektion von Lern-Eingabedaten als RBF-Zentren kann nicht garantieren, dass sie günstig verteilt sind. Da die SOM eine nachbarschaftserhaltende Abbildung liefert und viele *Visualisierungsmöglichkeiten* bietet, wurde sie dem K-Means-Algorithmus vorgezogen und in den Fallstudien angewendet.

Durch Grafiken sind viele Eigenschaften der Daten, der Prognosen und des Netzwerkverhaltens schneller und einfacher zu erkennen als durch die ebenfalls in Tabellenform in Textdateien gespeicherten Werte. Für die Lern- und Testdaten erzeugt *Visualrbfn* über die auch für andere Modelle generierbaren Zeitreihen- und Streudiagramme der Netzwerkausgaben \hat{y}_i , der Messwerte y_i und der Residuen — letztere zusätzlich auch mit einem Histogramm — hinaus noch andere Diagramme, welche die sonst schwierig zu überblickenden Netzausgaben den Messwerten anschaulich gegenüberstellen. Für das RBFSOM-

²⁾ Anschaulich bedeutet das, dass es an den Kanten der Voronoi-Polyeder keine Sprünge gibt [Roj93, S. 372].

Netzwerk lassen sich schon nach dem Training im Rahmen einer ersten Datenexploration durch U-Matrix-Darstellungen oder Sammon-Abbildungen der SOM-Kodebuchvektoren ungewöhnliche Strukturen in den Lerndaten erkennen. Die Werte ihrer einzelnen Komponenten sind auf dem SOM-Gitter durch Graustufen oder durch gleichartig wie das SOM-Gitter angeordneten Linien- oder Balkengrafiken zu veranschaulichen. Zusätzlich zu den Zeitreihengrafiken über den Nummern der Lern- und Testdaten und Streudiagrammen werden noch durch Graustufen-Darstellungen der Werte auf dem SOM-Gitter und damit der sonst verdeckten Schicht des RBFNs erzeugt. Extrapolationen sind durch die euklidischen Abstände zwischen Testeingabe- und den jeweils nächstliegenden Prototypen, die maximalen Aktivitäten aller RBF-Neuronen und für die Prototypen der Klassen die Likelihoods des Parzen-Dichteschätzers und des VI-Netzes auf der Basis von Schätzungen der Eingabe-Datendichte angezeigt, genauso wie sich die Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$ der zu einer Eingabe bezüglich der SOM ähnlichen Lern-Eingabedaten und die Spannweiten der ihnen zugehörigen y_i , die bedingten Mittelwerte $\bar{y}_{\kappa(\mathbf{x}_i)}$, die mittleren pro Klasse $\mathcal{X}_{\kappa(\mathbf{x}_i)}^L$ zu erwartenden Prognosefehler und schließlich nach einer Kreuzvalidierung die zu erwartenden Prognose- und Konfidenzintervalle durch die VI-Netz-Ausgaben und den Fehlerbalkenansatz für die Repräsentanten der einzelnen SOM-Klassen darstellen lassen. Für kleinere Testdatenumfänge können die lokal für die SOM-Klassen \mathcal{X}_k^L gegebenen Verteilungen der Ausgabegröße durch Box-Whisker-Plots beschrieben werden. Diese Intervalle sind aber nicht von der Güte der Anpassung des RBFNs abhängig. Die Varianz der Ausgabegröße innerhalb einer SOM-Klasse gibt nur einen eingeschränkten Aufschluss darüber, wie groß der zu erwartende Fehler der Prognosen des RBF SOM-Netzes sein wird. Es ist denkbar, dass das RBFN die zu approximierende Regressionsfunktion innerhalb einer solchen Klasse gut beschreibt, die dort aber eine große Änderungsrate aufweist. Dann wäre die Varianz auch bei einem kleinen Fehler hoch.

Zusätzlich lassen sich für die Messwerte der Test-Ausgabedaten, ggf. gemeinsam mit den Prognosen, durch Zeitreihendarstellungen mit grau hinterlegten Intervallen auf Kreuzvalidierungen basierende Fehlerbalken (Predictive-Error-Bars) und die beispielsweise durch die Minima $\min(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ und Maxima $\max(\mathcal{Y}_{\kappa(\mathbf{x}_i)}^L)$ der Ausgabegröße innerhalb der SOM-Klassen bezüglich der Lerndaten gegebenen Intervalle illustrieren. In der betreffenden Fallstudie lagen die meisten der Testdaten in den angegebenen Prognoseintervallen, vgl. Abb. 4.30 (S. 96) und Abb. 4.29 (S. 95).

Besonders wichtig ist die Möglichkeit der visuellen Inspektion der Neuronenaktivitäten in der verdeckten Schicht, die dadurch etwas von ihrem „Black-box-Charakter“ verliert. Die Veränderlichkeit innerhalb beliebiger Sequenzen von Eingabemustern werden durch die Trajektorien der Siegerneuronen auf dem SOM-Gitter deutlich, wobei zu bedenken ist, dass ein kleiner Schritt auf dem SOM-Gitter einen großen Schritt im Eingaberaum bedeuten kann und eine große Sprungweite von einem zum nächsten Siegerneuron nicht immer einen großen Unterschied zwischen den zugehörigen Eingaben repräsentiert, sondern dass auch die Graustufenunterschiede der U-Matrix-Darstellung in Betracht gezogen werden müssen. Durch die Verwendung des RBF SOM-Netzes lassen sich die maximalen Aktivitäten der RBF-Neuronen für jede beliebige Netzeingabe durch eine Graustufenre-

präsentation mit Hilfe des SOM-Gitters grafisch darstellen. Dadurch ist nicht nur erkennbar, ob eine Netzeingabe dem KNN unbekannt oder bekannt erscheint, sondern auch, welcher Teil der Schicht besonders erregt wird. Das kann u. a. dazu führen, dass auf diese Weise unzureichende Anpassungen des Netzes an die Daten oder Widersprüche in den Daten sichtbar werden, die sonst unentdeckt bleiben würden.

Überdies gewähren Darstellungen von zwei- und dreidimensionalen Graphen der durch das KNN vermittelten Approximation der Regressionsfunktion Einblicke in das zu erwartende Netzwerkverhalten, wobei sich die nicht variierten Netzeingabegrößen auf konstanten Werten halten lassen und eine oder zwei unabhängige Größen in den entsprechenden Wertebereichen der Lerndaten oder auch in beliebigen Intervallen variiert werden können. Bei der Voreinstellung der Simulationssoftware *Visualrbfn*, bei der für die Werte der konstant gehaltenen Größen die entsprechenden Mittelwerte verwendet werden, ist nicht in allen Anwendungen gesichert, dass dieser Vektor als gut bekannt eingestuft wird. Für partielle Funktionen mit nur einer sich verändernden und sonst konstanten Netzeingabe lassen sich für die auf diese Weise generierten Szenarien gleichzeitig die genannten Maße der lokalen Zuverlässigkeit illustrieren, z. B. die maximalen und die mittleren Aktivitäten der RBF-Neuronen, die Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sowie die Größe $\#\mathcal{X}_{\kappa(x)}^L$, die für die jeweiligen SOM-Neuronen-Einzugsgebiete bedingten Quantile sowie die erwarteten Fehlerbalken nach dem Predictive-Error-Bars-Ansatz und die Schätzungen der Konfidenz- und Prognoseintervalle durch die VI-Netzkomponenten. Für die Erzeugung und Interpretation von Darstellungen dieser Art existiert ein grundsätzliches Problem, falls die unabhängigen Größen nicht unabhängig voneinander sind. Dann lässt sich eine Größe nicht konstant halten, während eine andere verändert wird. Immerhin sollten das RBFSOM-Netzwerk und die ebenfalls implementierten Dichteschätzer das Verlassen der Punktwolke anzeigen. Die Durchführung einer Hauptkomponentenanalyse in der Datenvorverarbeitungsphase kann nicht nur die wichtige Funktion übernehmen, die Dimension des Eingaberaumes durch die Zusammenfassung redundanter Größen (Kompression) zu reduzieren, sondern auch voneinander unabhängige Größen erzeugen. Allerdings lassen sich die neu gewonnenen Prädiktoren (Faktoren) dann nicht immer leicht benennen und interpretieren. Die Veranschaulichung der (im weiteren Sinne partiellen) Netzausgabefunktion in Abhängigkeit von zwei veränderlichen und sonst konstanten Eingaben hat gegenüber den zweidimensionalen Graphen den Nachteil, dass sich die Funktionswerte nicht so leicht ablesen lassen und dass Zuverlässigkeitsmaße sich schwieriger darstellen lassen. Für die Darstellungen der dreidimensionalen Graphen könnten in zukünftigen Software-Versionen (etwa unter Verwendung von *Mathematica*) die als Gittermodell dargestellte Oberflächen oder die Grundflächen entsprechend eingefärbt oder weitere Grafiken dreidimensionaler Graphen der jeweiligen Funktionen, welche die Dichten, Streuungen und erwarteten Fehler angeben, erzeugt werden.

Bei allen Grafiken, vor allem der verdeckten Schicht, ist zu bedenken, dass das Ergebnis des Trainings vieler KNNs durch den Zufall bedingt ist und dass daher selbst bei gleicher Wahl der Parameter und den selben Lerndaten zumindest bezüglich Drehungen und Spiegelungen unterschiedliche Darstellungen der SOM entstehen.

Die Hybridisierung der Netze vereint nicht nur die Vorteile der einzelnen Komponenten, sondern bringt auch einige prinzipielle und praktische Probleme mit sich. Grundsätzlich muss bei RBF-SOM-Netzen die optimale Größe der SOM zur Partitionierung des Eingaberaumes nicht der optimalen Anzahl von RBF-Zentren zur Approximation der Regressionsfunktion entsprechen und die globale Prognosegüte des RBF-SOM-Netzes mit der unüberwacht trainierten RBF-Schicht ist möglicherweise schlechter als von ausschließlich überwacht trainierten Konkurrenten, vgl. [Bis95, S. 190]. Hier müssen Kompromisse eingegangen werden. Nach der Bestimmung einer günstigen Anzahl von RBF-Zentren und damit SOM-Zellen, beispielsweise durch wiederholte (Kreuz-)Validierungen sind die Architektur (Dimension, Höhe, Breite) und die Trainingsparameter der SOM festzulegen. Dabei ist keineswegs gesichert, dass die SOM nach dem Lernen die Punktwolke der Lern-Eingabedaten gut beschreibt. Für Visualisierungszwecke sind ein- und vor allem zweidimensionale SOM-Gitter besonders geeignet. Allerdings lassen sich Daten aus hochdimensionalen Räumen nicht immer günstig in zwei Dimensionen abbilden. Eine Vorstellung von der prinzipiellen Schwierigkeit der SOM-Repräsentation einer Punktwolke in einem höher dimensionalen Raum (z. B. \mathbb{R}^2) durch eine SOM, deren Gitter in einem niedrig dimensional Raum eingebettet ist (hier \mathbb{R}), vermittelt Abb. 5.1.³⁾ Nebenbei

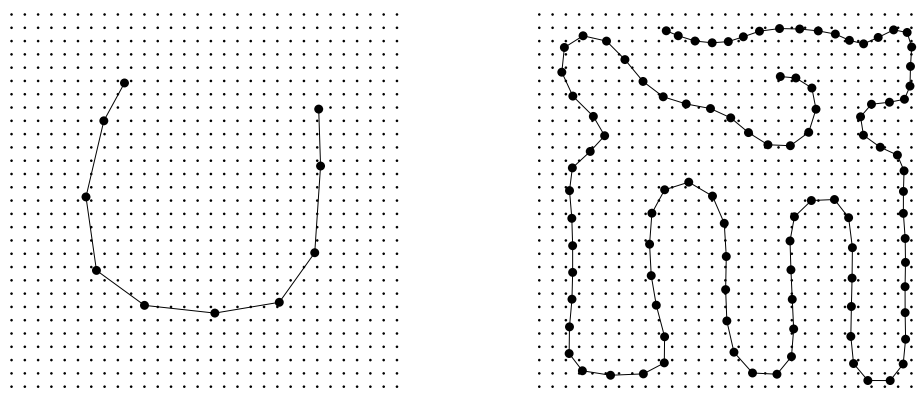


Abbildung 5.1: Schwierigkeiten bei der Repräsentation eines intrinsisch höherdimensionalen Raumes durch Kodebuchvektoren in SOM-Gittern niedrigerer Dimension. Ketten aus 10 (links) und 100 (rechts) SOM-Neuronen sollen ein Gitter aus 30×30 gleichmäßig in einem Quadrat verteilten Punkten repräsentieren.

wird durch die Grafik deutlich, wie es zur Aktivierung mehrerer auf der Karte bzw. Kette voneinander entfernt liegender SOM- und RBF-Neuronen kommen kann, deren Kodebuchvektoren bzw. RBF-Zentren im Eingabedatenraum nahe beieinander liegen. Die

³⁾ Bei einer genügend großen Anzahl von SOM-Neuronen kann die Darstellung einer *Peano-Kurve* ähneln, vgl. [RMS90, S. 81] und [Koh01, S. 114]. Zur Erstellung dieser Grafiken wurden die jeweils 900, im Intervall $[0,1]^2$ liegenden Lerndatenpunkte sowie die mit *SOM_PAK* positionierten Kodebuchvektoren, die nach dem Training der SOMs mit 20'000 Lernschritten für die 10 und 300'000 Lernschritten für die 100 SOM-Zellen und Lernraten von anfangs $\eta = 0.8$ vorlagen, aus den jeweiligen Dateien mit einem Texteditor in PostScript dargestellt. Es handelt sich dabei nicht um Sammon-Abbildungen.

SOM kann die Lern-Eingabedaten aber auch durch zu kurzes oder aufgrund ungünstiger Parameterwahl ungenügendes Training schlecht repräsentieren. Gegebenenfalls ist das Lernen des SOM-Netzes daher mit unterschiedlichen Parametern mehrfach zu wiederholen. Aber auch für das RBF-Netz sind Designparameter wie die Anzahl von Nachbarn oder die absolute Größe der Bandbreiten etwa auf der Basis von Validierungsdaten (ggf. auch Kreuzvalidierungen) möglichst günstig zu wählen.

Einige Probleme ergeben sich aus der Wahl der verwendeten Netztypen, deren Aktivität von dem (euklidischen) Abstand der Datenpunkte von den Referenzvektoren bzw. RBF-Zentren abhängt. Um dem „Fluch der Dimensionen“ zu entkommen, sollte eine Dimensionsreduzierung in der Vorverarbeitungsphase oder als Bestandteil des Netzwerktrainings vorgenommen werden.

Redundante Prädiktoren können durch die Anwendung von Hauptkomponentenanalysen oder von Flaschenhalsnetzen zusammengefasst werden. Erstere können überdies voneinander unabhängige Größen liefern, was die vorgestellten Veranschaulichungsmöglichkeiten partieller Funktionen und die Abschätzung der Wichtigkeit der einzelnen Komponenten vereinfacht. In der vorliegenden Arbeit kamen diese Verfahren nicht zum Einsatz, um eine möglichst anschauliche Interpretation der Ergebnisse zu gewährleisten⁴⁾ und um den Informationsverlust gering zu halten, der die im Regressionsmodell abhängige Größe nicht berücksichtigt.

Eine ebenfalls ohne Blick auf die Ausgabegröße gerichtete Möglichkeit besteht darin, mit Hilfe einer Korrelations- oder einer Clusteranalyse auf der Basis der (euklidischen) Abstände der Vektoren aus den Werten der einzelnen Prädiktoren redundante Eingabegrößen zu entfernen. Dieser Ansatz lässt sich auch für potentielle Ausgabegrößen anwenden, wie das Beispiel der Anzahl der in einer Emergenzfall über 30 Jahre gefangenen Insekten verschiedener EPT-Arten in Abschnitt 4.1 zeigt, wobei die Ähnlichkeiten durch eine SOM und eine Sammon-Abbildung auf eine übersichtliche Art deutlich wurden. Die überwachte Selektion der für die Bildung eines Regressionsmodells wichtigsten Prädiktoren kann mit Hilfe der den RBFNs sehr ähnlichen, aber durch ein Einschnitt-Training noch schneller lernenden GRNNs erfolgend, die mit einem schrittweisen Hinzunehmen oder Entfernen von Prädiktoren oder durch Genetische Algorithmen viele Kombinationen von Eingabegrößen validieren. Dabei empfiehlt sich die Verwendung der gleichen Basisfunktionen für die verwandten Netztypen.

Für ein bereits bestehendes, mit der Software *Visualrbfn* simuliertes RBFN lassen sich — im Gegensatz zum Entfernen des Prädiktors wie beim der Software *GRNN* mit den gleichnamigen Netzwerken — ohne ein Neutraining Aussagen über die Redundanz und Relevanz der einzelnen Prädiktoren treffen, indem jeweils eine unabhängige Größe auf einem festen Wert (etwa dem Mittelwert) gehalten und dann ein Validierungsfehler ermittelt wird. Für nicht überflüssige, wichtige Größen sollte sich dadurch ein deutlich erhöhter Fehler ergeben.

⁴⁾ Die gefundenen Hauptkomponenten sind nicht immer leicht zu interpretieren. Es ist kaum zu erwarten, dass die Komponenten, die mit „kurzfristiger Niederschlag“ und „aktuelle Temperatur“ zu bezeichnen wären, orthogonal zueinander stehen.

In den Fällen, in denen die Prognosegüte wichtiger als die Visualisierungen und die Abschätzungen der Zuverlässigkeit, kann das Gradientenabstiegsverfahren auf der Basis der Fehlerrückführung im Sinne eines Nachtrainings der vorher durch hybrides Training angepassten RBFNs angewendet werden. Jede Skalierung ist willkürlich. Die Varianz- oder Spannweiten-angleichenden Skalentransformationen wie die z-Transformation oder die [0,1]-Skalierung können als günstig aufgefasst werden, wenn die Eingabegrößen gleich wichtig für die Vorhersage der Ausgabevariablen eingeschätzt werden. Ansonsten kann eine Reskalierung eine Verbesserung der Generalisierungsfähigkeit des Regressionsmodells ermöglichen, da sich die Komponenten des RBFNs auf die wichtigen Prädiktoren konzentrieren können. Bei betragsmäßig sehr wenig ins Gewicht fallenden Skalierungsfaktoren lassen sich die entsprechenden, kaum relevanten Prädiktoren sogar aus dem Modell entfernen, woraus kleinere und damit effizienter lernende und arbeitende Netzwerke resultieren. Der in der Software *Visualrbfn* implementierte Ansatz durch Fehlerrückführung und Gradientenabstiegsverfahren erwies sich im einfachen, artifiziellen Spezialfall als überzeugend. Sein Einsatz bei der Anwendung auf Abflussprognosen war ebenfalls erfolgreich. Durch das verwendete Optimierungsverfahren werden die Bandbreiten der RBFs nicht einzeln verändert. Zwar könnte man versuchen, diese ebenfalls durch das Gradientenabstiegsverfahren zu optimieren, allerdings zeigen die in [Bis95, S. 191] zitierten Ergebnisse, dass dann die Bandbreiten in vielen Fällen sehr groß werden können. Ob durch das *Levenberg-Marquardt-Verfahren*, wie dies beispielsweise in [CI00] zur Anpassung der Gewichte eines hybriden Netzes mit RBF-Zellen und Perzeptron-Neuronen verwendet wurde, und eine Umsetzung der Verfahren mit den Programmiersprachen C oder C++ eine wesentliche Beschleunigung der Parameteroptimierung mit sich bringt, bleibt zu überprüfen.

Bei diesem reskalierenden und den vorher genannten komprimierenden oder selektierenden Verfahren ist zu bedenken, dass ein Prädiktor in bestimmten Intervallen seines Wertebereiches möglicherweise wichtiger als in anderen sein kann. Ein Beispiel für die Abhängigkeit der Relevanz einer Größe von ihren Werten ist die Lufttemperatur in der Nähe des Gefrierpunktes des Wassers und wiederum bei sehr hohen Werten für die Abflusswirksamkeit von Niederschlägen.

Der Umfang der für das Lernen oder Testen des RBFSOM-Netzes und die Ausführung der Simulationssoftware verwendeten Stichprobe ist von großer Bedeutung. Bei einer kleinen Anzahl von Lerndatensätzen lassen sich die Aussagen des KNNs kaum untermauern. Bei der in Abschnitt 4.1 dokumentierten Fallstudie hatte allein die Zuordnung der Datensätze zum Training oder dem Testen der Generalisierungsfähigkeit des RBFSOM-Netzes einen großen Einfluss. Wäre etwa zufällig der Messwert für das letzte Jahr des Untersuchungszeitraumes (1998) als Testdatensatz verwendet worden, bei dem die Abundanz von *A. fimbriata* in der Emergenzfalle ihr Minimum annahm, wäre die Streuung der Ausgabegröße in der Klasse 2 wesentlich geringer gewesen und für diese Prognose hätte man einen sehr großen Fehler erzeugt. Bei Abb. 4.6 ist fraglich, wie zuverlässig die Schätzungen der lokal zu erwartenden Zuverlässigkeit bei so wenigen Daten sein können. Für sehr kleine Datenumfänge wie in diesem Fall sollte man für die Abschätzung der Generali-

sierungsfähigkeit des KNNs am besten eine Kreuzvalidierung vornehmen, um jeweils so viele Datenpunkte wie möglich für das Training verwenden zu können.

Große Stichprobenumfänge führen hingegen eher zu praktischen Problemen bei der Simulation der Netzwerke am Computer. Dabei ist an erster Stelle eine langwierige Programmausführung zu nennen, da viele verdeckte Neuronen (SOM- bzw. RBF-Neuronen) und zahlreiche Auswertungen der Gauß-Funktion nötig sind. Das gilt insbesondere für die zusätzliche Anwendung des Parzen-Dichteschätzers, bei dem für jedes Element eines Testdatensatzes für jeden einzelnen Trainingsdatensatz eine Gaußfunktion berechnet werden muss. In noch stärkerem Maße sind große Datenmengen für vielfach wiederholte Berechnungen von Netzwerkausgaben, etwa bei der Bestimmung relevanter Prädiktoren, der automatischen Reskalierung durch Gradientenabstiegsverfahren und Kreuzvalidierungen. In solchen Fällen ist der Umfang der Lerndaten etwa durch eine Zufallsauswahl zu verringern oder die Daten-reduzierende Vorverarbeitung anzuwenden, die in *Visualrbfn* implementiert ist.

Die am Anfang dieser Arbeit an ein Prognosesystem gerichteten Fragen lassen sich mit den vorgestellten Ansätzen weitgehend beantworten. Die Anzahl ähnlicher Messungen, auf die sich eine Prognose stützen kann, wird durch die Anzahl der im selben Einzugsgebiet liegenden Lerndaten beschrieben. Die Ähnlichkeit einer Netzeingabe als Grundlage für eine Prognose zu anderen Daten ist durch den Quantisierungsfehler als minimalen Abstand zum nächsten SOM-Kodebuchvektor gegeben und der Grad der Neuartigkeit lässt sich zusätzlich durch die Maxima lokaler RBFs sowie die Likelihoods der VI-Netz- und Parzen-Dichteschätzer bestimmen. Die Spannweiten (ggf. auch andere Quantile) der vorherzusagenden Größe für Einzugsgebiete von SOM-Neuronen spiegeln näherungsweise ihre lokale Streuung wider. Den lokal zu erwartenden Fehler können die mittleren Prognosefehler in jeder SOM-Klasse sowie Predictive-Error-Bars schätzen. VI-Netzkomponenten geben Vertrauens- und Vorhersageintervalle an. Für alle genannten Größen und die geschätzten Regressionen wurden grafische Veranschaulichungen für Lern- und Testdaten, Prototypen der Klassen, einzelne Eingaben und Szenarien für die Ausgabegröße mit ein oder zwei unabhängigen Veränderlichen und sonst konstanten Netzeingaben vorgestellt.

Damit ist es in der vorliegenden Dissertation gelungen, verschiedene Neuronale Netze zu hybridisieren, die Resultate in eine Simulationssoftware umzusetzen und ihre vielseitige Anwendbarkeit exemplarisch in Fallstudien zu zeigen, wobei für diese adaptiven Systeme — anders als für die meisten Wahrsager, Wetterfrösche, Börsen-Gurus und viele andere KNN-Typen — Angaben über die zu erwartende Zuverlässigkeit von Prognosen vorliegen und die Arbeitsweise des KNNs auch durch visuelle Inspektion sonst verborgener Komponenten transparenter wird.

6 Anhang

6.1 Artnamen und Abundanzmuster der untersuchten Eintags-, Stein- und Köcherfliegen

Tabelle 6.1: Artnamen aus den Insektenordnungen der Ephemeroptera (Eintagsfliegen), Plecoptera (Steinfliegen) und Trichoptera (Köcherfliegen), deren Populationschwankungen am Breitenbach in dieser Arbeit untersucht wurden.

Abkürzung	Artbezeichnung	Ordnung
A.fim	<i>Apatania fimbriata</i> (PICTET)	Trichoptera
A.fus	<i>Agapetus fuscipes</i> CURTIS	Trichoptera
A.red	<i>Adicella reducta</i> MACLACHLAN	Trichoptera
A.sta	<i>Amphinemura standfussi</i> RIS	Plecoptera
B.rho	<i>Baetis rhodani</i> PICTET	Ephemeroptera
B.ver	<i>Baetis vernus</i> CURTIS	Ephemeroptera
C.lut	<i>Centroptilum luteolum</i> MÜLLER	Ephemeroptera
C.vil	<i>Chaetopteryx villosa</i> FABRICIUS	Trichoptera
D.ann	<i>Drusus annulatus</i> STEPHENS	Trichoptera
E.dan	<i>Ephemera danica</i> MÜLLER	Ephemeroptera
E.ign	<i>Ephemerella ignita</i> PODA	Ephemeroptera
E.muc	<i>Ephemerella mucronata</i> BENGTTSSON	Ephemeroptera
E.ven	<i>Ecdyonurus venosus</i> FABRICIUS	Ephemeroptera
H.dig	<i>Halesus digitatus</i> SCHRANK	Trichoptera
H.ins	<i>Hydropsyche instabilis</i> CURTIS	Trichoptera
H.sax	<i>Hydropsyche saxonica</i> MACLACHLAN	Trichoptera
I.goe	<i>Isoperla goertzi</i> ILLIES	Plecoptera
L.dig	<i>Leuctra digitata</i> KEMPNÝ	Plecoptera
L.nig	<i>Leuctra nigra</i> OLDENBERG	Plecoptera
L.pri	<i>Leuctra prima</i> KEMPNÝ	Plecoptera
M.lon	<i>Micrasema longulum</i> MACLACHLAN	Trichoptera
N.cam	<i>Nemoura cambrica</i> STEPHENS	Plecoptera
N.cin	<i>Nemoura cinerea</i> RETZIUS	Plecoptera
N.fle	<i>Nemoura flexuosa</i> AUBERT	Plecoptera
N.mar	<i>Nemoura marginata</i> PICTET	Plecoptera

N.pic	<i>Nemurella pictetii</i> KLAPÁLEK	Plecoptera
O.alb	<i>Odontocerum albicorne</i> SCOPOLI	Trichoptera
P.aub	<i>Protonemura auberti</i> ILLIES	Plecoptera
P.cin	<i>Potamophylax cingulatus</i> STEPHENS	Trichoptera
P.con	<i>Plectrocnemia conspersa</i> CURTIS	Trichoptera
P.int	<i>Protonemura intricata</i> RIS	Plecoptera
P.luc	<i>Potamophylax luctuosus</i> PILL & MITT.	Trichoptera
P.nit	<i>Protonemura nitida</i> PICTET	Plecoptera
P.sub	<i>Paraleptophlebia submarginata</i> STEPHENS	Ephemeroptera
R.fas	<i>Rhyacophila fasciata</i> HAGEN	Trichoptera
S.pal	<i>Silo pallipes</i> FABRICIUS	Trichoptera
S.per	<i>Sericostoma personatum</i> KIRBY & SPENCE	Trichoptera
S.tor	<i>Siphonoperla torrentium</i> PICTET	Plecoptera
T.ros	<i>Tinodes rostocki</i> MACLACHLAN	Trichoptera
W.occ	<i>Wormaldia occipitalis</i> PICTET	Trichoptera

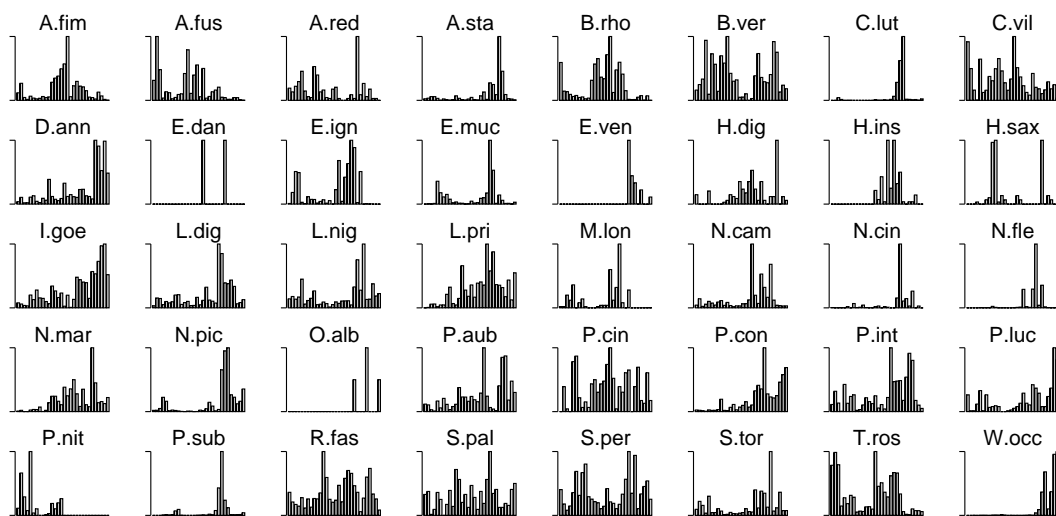


Abbildung 6.1: Muster jährlicher, relativierter Abundanzen von 40 ausgewählten EPT-Arten in der Emergenzfalle „Haus B“ im Untersuchungszeitraum 1969–1998; die Abkürzungen der Artnamen sind in Tab. 6.1 erklärt.

6.2 Optionen der Simulationssoftware

Nachfolgend ist der Inhalt der Datei `optionscheck.m` mit den gewählten und vorgegebenen Programmoptionen von *Visualrbfn* wiedergegeben, wie sie in dem Abschnitt 4.2 für die Prognose des täglichen Abflusses des Breitenbaches an „Haus B“ in der Untersuchung ohne Informationen über vorhergehende Abflusswerte als Netzeingaben verwendet wurden, vgl. Unterabschnitt 3.1.4 ab S. 40.

# Option name	Current value	Default value	Modified
BPContinue	= false	; # false	
BPetaS	= 0.001	; # 0.001	
BPetaW	= 0.001	; # 0.001	
BPetaZ	= 0.001	; # 0.001	
BPLogStepNumber	= 3	; # 3	
BPMomentumB	= 0.8	; # 0.8	
BPNumSteps	= 25	; # 25	
BPStopRMSE	= 0.005	; # 0.005	
BPSVD	= true	; # true	
CVCrossValidate	= true	; # true	
CVNumValSamples	= 450	; # 5	# !!!
CVRandomOrder	= false	; # true	# !!!
CVRatioValSamples	= 0	; # 0	
CVReuse	= false	; # false	
CVShowProgress	= true	; # true	
DataFileName	= "lern.dat"	; # "learn.dat"	# !!!
DataInputColumns	= [1,2,3,4,5,6,7]	; # 0	# !!!
DataOutputColumns	= 8	; # 0	# !!!
DataRescaleInput	= false	; # false	
Mode	= 1	; # 1	
NDEF	= 12345	; # 12345	
ParzenBandWidth	= 0	; # 0	
ParzenDataFile	= "l_som_data.dat"	; # "l_som_data.dat"	
ParzenDensity	= false	; # false	
PlotActivities	= false	; # false	
PlotGraphics	= true	; # true	
Quiet	= false	; # false	
RBFBandWidthConst	= false	; # false	
RBFBandWidthDefault	= 0.4	; # 0.4	
RBFBandWidthNN	= 2	; # 8	# !!!
RBFBandWidthStretch	= 2.5	; # 1	# !!!
RBFCalcLearnError	= true	; # true	
RBFCentresNum	= 78	; # 24	# !!!
RBFCentresPos	= 4	; # 4	
RBFEPESupport	= true	; # false	# !!!
RBFType	= "Gauss"	; # "Gauss"	
RBFVISupport	= true	; # false	# !!!
RBFWithBias	= true	; # true	
RBFWithLinearTerm	= false	; # false	
ReduceData	= false	; # false	
ReduceDataMinDist	= 0.05	; # 0.05	
SOMAnalyze	= true	; # true	
SOMCBVFileName	= "l_som_cbv.smf"	; # "l_som_cbv.smf"	
SOMHeight	= 0	; # 0	
SOMNumTrainingSteps	= 100000	; # 100000	
SOMReuse	= true	; # false	# !!!
SOMWidth	= 0	; # 0	
VerboseLevel	= 3	; # 3	
Visu3DColX1	= 1	; # 1	
Visu3DColX2	= 1	; # 2	# !!!
Visu3DConstVector	= [0.2,0.3,0.2,0.3,0.8,0.9,0.6]	; # 0	# !!!
Visu3DMaxX1	= 0	; # 0	

```

Visu3DMaxX2      = 0                ; # 0
Visu3DMeshLabelX1 = "x_1"              ; # " " # !!!
Visu3DMeshLabelX2 = "x_4"              ; # " " # !!!
Visu3DMeshLabelY  = "Abfluss, [0,1]-skaliert" ; # " " # !!!
Visu3DMinX1       = 0                ; # 0
Visu3DMinX2       = 0                ; # 0
Visu3DNumGridPoints = 20            ; # 15 # !!!

```

6.3 Abkürzungen

Neben den in wissenschaftlichen Arbeiten üblichen Abkürzungen wie z. B. (zum Beispiel), d. h. (das heißt), u. a. (unter anderem), ggf. (gegebenenfalls), Abb. (Abbildung), Tab. (Tabelle), Gl. (Gleichung), S. (Seite), f. (folgende Seite), ff. (folgende Seiten), bzw. (beziehungsweise), sog. (sogenannt), vgl. (vergleiche), usw. (und so weiter) werden in der vorliegenden Arbeit noch folgende Abkürzungen verwendet:

ASCII	—	American Standard Code for Information Interchange
EPS	—	Encapsulated PostScript
EPT	—	Eintags-, Stein- und Köcherfliegen
GRNN	—	General Regression Neural Network
GUI	—	grafische Benutzerschnittstelle
KNN	—	Künstliches Neuronales Netzwerk
LNN	—	Lineares Neuronales Netzwerk
LVQ	—	Lernende Vektorquantisierung
MLP	—	Multi-Layer-Perzeptron
PCA	—	Hauptkomponentenanalyse
PRBFN	—	Perceptron Radial Basis Net
RBF	—	radiale Basisfunktion
RBFN	—	Radiale-Basisfunktionen-Netzwerk
RBF SOM	—	Hybrid aus RBF- und SOM-Netz
RMSE	—	Wurzel der mittleren Fehlerquadratsumme
SOM	—	selbstorganisierende Karte
TDS-Nr.	—	Testdatensatz-Nummer
U-Matrix	—	Unified Distance Matrix
VI-Netz	—	Validity-Index-Netzwerk

6.4 Häufig verwendete Symbole

c	—	Index des Siegerneurons einer SOM (S. 11)
d	—	Eingabedimension (S. 16)
(e)	—	Kennzeichnung von Begriffen in der deutsch-englischen Wortliste ab S. 126
E	—	Fehlerfunktion (S. 31)
h, h_k	—	Bandbreite radialer Funktionen (S. 17)
i	—	Index für Daten $(1, \dots, n)$
j	—	Index für Dimensionen bzw. Eingabeneuronen $(1, \dots, d)$
k	—	Index für SOM-Neuronen und ihre Kodebuchvektoren sowie RBF-Zentren $(1, \dots, m)$
m	—	Anzahl von SOM- oder RBF-Neuronen in einem KNN
n	—	Anzahl von Daten bzw. Lern- oder Test-Daten (Stichprobenumfang)
\mathbb{R}	—	Menge der reellen Zahlen
\mathbf{s}	—	Lerndaten-Muster bei einem SOM-Lernschritt („Stimulus“) (S. 12)
t	—	Lernschrittzähler beim SOM-Training und Gradientenabstiegsverfahren sowie Zeitparameter bei Zeitreihen
w_k	—	Gewicht des Ausgabeneurons für RBF-Neuronen-Ausgänge und den Bias in RBFNs (S. 17)
\acute{w}_j	—	Gewicht des Ausgabeneurons für eine Verbindung zum Eingabeneuron $j, j = 1, \dots, d$ (S. 18)
\mathbf{X}, \mathbf{x}	—	Prädiktor und seine mögliche Realisierung $\mathbf{x} \in \mathbb{R}^d$ (S. 6)
X, x, x_i	—	wie $\mathbf{X}, \mathbf{x}, \mathbf{x}_i$ für $d = 1$ (S. 2)
\mathbf{x}_i	—	Messwert der Eingabevariablen, ggf. reskaliert (S. 6)
\mathcal{X}_k	—	Einzugsgebiet des SOM-Neurons \mathbf{z}_k (S. 11)
\mathcal{X}_k^L	—	Menge der Lerndaten \mathbf{x}_i in \mathcal{X}_k (S. 12)
$\#\mathcal{X}_k^L$	—	Anzahl der Elemente von \mathcal{X}_k^L (S. 12)
Y, y	—	abhängige Größe und ihre mögliche Realisierung (S. 2)
y_i	—	Messwert der Ausgabevariablen, ggf. reskaliert (S. 2)
\hat{y}	—	geschätzte Regressionsfunktion, $\hat{y}_i = \hat{y}(\mathbf{x}_i)$ (S. 2)
\bar{y}	—	Mittelwert über alle Daten y_i für $i = 1, \dots, n$
\bar{y}_k	—	Mittelwert für alle $y_i \in \mathcal{Y}_k^L$ (S. 6)
\mathcal{Y}_k^L	—	Menge der Lerndaten y_i , für die \mathbf{x}_i in \mathcal{X}_k liegt (S. 27)
\mathbf{z}_k	—	Kodebuchvektor (S. 11) bzw. RBF-Zentrum (S. 17) im \mathbb{R}^d
η	—	Lernkonstante für das SOM-Training (S. 13)
η_s, η_z, η_w	—	Lernkonstanten für den Gradientenabstieg (S. 31)
ϕ	—	Basisfunktion bei RBFNs (S. 17)
$\kappa(\mathbf{x})$	—	Index des Siegerneurons einer SOM für \mathbf{x} (S. 11)
μ	—	Konstante im Momentum-Term beim Gradientenabstieg (S. 31)
ψ	—	SOM-Nachbarschaftsfunktion (S. 13)

6.5 Auswahl englischer Fachbegriffe

Um Missverständnisse beim Studium der Fachliteratur zu vermeiden und um dem Leser Suchbegriffe für Nachforschungen im Internet zu nennen, sind im Folgenden englische Fachtermini angegeben, deren deutsche Entsprechungen in dieser Dissertation verwendet wurden. Englische Wörter, die sich vor allem im Bereich der Computertechnologie in der deutschen Sprache durchgesetzt haben, z. B. „Software“, „Shell“ und „Link“, wurden nicht übersetzt.

Weitere Auswahlen englischer Fachausdrücke aus der Statistik und ihre deutschen Übersetzungen finden sich in [KSV92, S. 283 ff.] und [Kre68, S. 370 ff.], ein englisch-deutsches Register ökologischer Begriffe ist in [ST83, S. 313 ff.] gedruckt. Ein kleines Fachwörterverzeichnis zum Thema Neuronale Netzwerke ist in [Hof93, S. 169 ff.] enthalten. Außerdem sei auf die wachsenden Möglichkeiten von „Online-Wörterbüchern“ im Internet hingewiesen.

Da im Stichwortverzeichnis englischsprachige Begriffe ihren deutschen Übersetzungen zugeordnet werden, ist an dieser Stelle nur eine Tabelle zu finden.

a-posteriori-Wahrscheinlichkeitsdichte	—	posterior probability density
a-priori-Wahrscheinlichkeitsdichte	—	prior probability density
Abbildung	—	mapping
Abfluss	—	discharge, flow, runoff
Abundanz	—	abundance
Ausdehnung (einer Basisfunktion)	—	deviation
Ausgabe	—	output
Ausgabeschicht	—	output layer
bayesisches Netzwerk	—	Bayesian network
Bestimmtheitsmaß	—	determination coefficient
Breite (einer Basisfunktion)	—	width
Eingabe	—	input
Eingabeschicht	—	input layer
Einschrittprognose	—	one step ahead prediction
exponentielles Glätten	—	exponential smoothing
fehlender Messwert	—	missing value
Fehlerbalken	—	error bars
Fehlerquadratsumme	—	sum-of-squared errors
Fehlerrückführung	—	backpropagation of errors
Fluch der Dimensionen	—	curse of dimensionality
gaußsche Basisfunktion	—	Gaussian basis function
Gitter	—	grid, lattice
grafische Benutzerschnittstelle	—	graphical user interface

Hauptkomponentenanalyse	—	principal components analysis
Hesse-Matrix	—	Hessian matrix
Hochsprache	—	high-level language
Kohonen topographische Merkmalskarte	—	Kohonen topographic feature map
Komitee	—	committee
Komponentenebene	—	component plane
Konfidenzintervall	—	confidence interval
Kreuzvalidierung	—	(<i>S</i> -fold) cross-validation
Künstliches Neuronales Netzwerk	—	artificial neural network
Lebensgemeinschaft	—	community
Lernen	—	learning
Muster	—	pattern
Nachbarschaftsfunktion	—	neighbourhood kernel
nächster Nachbar	—	nearest neighbo(u)r
Neuartigkeit	—	novelty
Prognose	—	prediction
Prognose-Intervall	—	prediction interval
radiale Basisfunktion	—	radial basis function
Referenzaufruf	—	call-by-reference
Rohdaten	—	raw data
Sammon-Abbildung	—	Sammon mapping
Schicht (eines KNNs)	—	layer
selbstorganisierende Karte	—	self-organizing feature map
Streudiagramm	—	scatter plot
Überanpassung	—	overfitting
überwachtes Lernen	—	supervised learning
unüberwachtes Lernen	—	unsupervised learning
Vektorquantisierung	—	Learning Vector-Quantization
verdeckte Schicht	—	hidden layer
Voronoi-Parkettierung	—	Voronoi tessellation
Vorverarbeitung	—	pre-processing
vorwärts gerichtetes Netzwerk	—	feed-forward network
Wurzel der mittleren Fehlerquadratsumme	—	root mean squared error
Zeiger	—	pointer
Ziehen mit Zurücklegen	—	resampling with replacement
Zielgröße	—	target

6.6 Software und Bezugsquellen

In der folgenden Auflistung finden sich zu der in der vorliegenden Dissertation erwähnten Software (mit Ausnahme der Betriebssysteme und als allgemein bekannt vorausgesetzter Standardsoftware) der Name, die verwendete Version, die Namen der Autoren bzw. Hersteller sowie eine „Download“-Möglichkeit oder eine Informationsmöglichkeit im Internet, auf der u. a. eine Bezugsadresse angegeben ist. Die Angaben wurden zuletzt am 31. März 2003 überprüft.

<i>Cygwin</i>	Version der cygwin1.dll: 1.3.22-1; Red Hat, Inc.; http://www.cygwin.com
<i>Gnuplot</i>	Version: 3.7 (Patchlevel 0); Thomas WILLIAMS, Colin KELLEY, Russell LANG, Dave KOTZ, John CAMPBELL, Gershon ELBER u. a.; ftp://ftp.dartmouth.edu/pub/gnuplot
<i>GRNN</i>	Simulation von <i>General-Regression-Neural-Networks</i> und verschiedenen Verfahren zur Selektion relevanter Prädiktoren; Version vom 23.05.2000; Michael OBACH; http://www.neuro.informatik.uni-kassel.de/cgi-bin/HTMLManPage?grnn
<i>Mathematica</i>	Version: 4.0; Wolfram Research, Inc.; http://www.wolfram.com
<i>Matlab</i>	Version: 13; The MathWorks, Inc.; http://www.mathworks.com
N^2E^4	Forschungsgruppe Neuronale Netzwerke, Universität Kassel; http://www.neuro.informatik.uni-kassel.de
<i>Octave</i>	GNU Octave; Version: 2.1.31 (i586-pc-cygwin); John W. EATON u. a.; http://www.octave.org
<i>Ploticus</i>	Version: 2.03 (win32); Stephen C. GRUBB; http://ploticus.sourceforge.net
<i>SOM_PAK</i>	Self-Organizing Map Program Package; Version: 3.1 (7. April 1995); SOM Programming Team of the Helsinki University of Technology, Laboratory of Computer and Information Science (Rakentajanaukio 2 C, SF-02150 Espoo, Finnland); Teuvo KOHONEN, Jussi HYNNINEN, Jari KANGAS, Jorma LAAKSONEN; ftp://cochlea.hut.fi:/pub/som_pak
<i>SPSS</i>	SPSS for Windows; Version: 10; SPSS Inc.; http://www.spss.com
<i>Trajan</i>	Neural Network Simulator; Version: 6.0D; Trajan Software Ltd.; http://www.trajan-software.demon.co.uk
<i>Visualrbfn</i>	Simulation von RBF- und RBFSOM-Netzwerken mit Erweiterungen; Version: 4.5; Michael OBACH; http://www.Michael-Obach.de

6.7 Abbildungsverzeichnis

1.1	Datenpunkte, bedingter Erwartungswert, Modell und Abweichung von Modell- und Messwert	2
1.2	Punktewolke mit Regressionsmodell und als konstant angenommener Fehlerfunktion	4
1.3	Punktewolke mit Modell und nicht-konstantem Prognoseintervall sowie Dichteschätzung	5
1.4	Partitionierung des Eingabedatenraumes und Berechnung einfacher lokaler Modellgütegrößen für die einzelnen Klassen	5
1.5	Superposition von radialen Basisfunktionen.	6
2.1	Schematischer Aufbau eines vorwärts gerichteten Schichtennetzwerkes. .	10
2.2	Beispiel einer Voronoi-Parkettierung der Ebene durch zufällig ausgewählte Punkte	12
2.3	Lernen einer SOM I: Initialisierung des Neuronengitters	13
2.4	Lernen einer SOM II: Bestimmung des Siegers	14
2.5	Lernen einer SOM III: Adaptionsschritt	15
2.6	Veranschaulichung der Superposition von gaußschen Basisfunktionen, einem linearen Anteil und einem Bias-Element	19
2.7	Problem der Unter- und Überanpassung	24
2.8	Veranschaulichung der auf einem Komitee oder Ensemble von Modellen basierenden Prognose und ihrer Streuungen	25
2.9	Voronoi-Parkettierung des Eingaberaumes und Überlagerung von dreidimensionalen „Gauß-Glocken“	27
2.10	Approximation einer sinusoiden Funktion durch RBFNs mit und ohne vorgeschalteter Reskalierungsschicht	32
2.11	Vergleich der Approximation einer affinen Funktion durch ein RBFN mit automatischer Reskalierung der Eingabegröße und ein RBFSOM-Netz . .	34
4.1	Jährliche Abundanzen ausgewählter EPT-Arten	69
4.2	Sammon-Abbildung von Kodebuchvektoren und jährlichen Abundanzmustern von EPT-Arten über 30 Jahre an „Haus B“	70
4.3	Visualisierung der Abhängigkeit der jährlichen Abundanz von <i>Tinodes rostocki</i> von je zwei Variablen auf der Basis eines RBFSOM-Netzwerk-Modells	71
4.4	Vorhersage der jährlichen Abundanzen zweier EPT-Arten für sechs Jahre .	72
4.5	Prototypen für Klassen von Abflussmustern während des Lebenszyklus von <i>Apatania fimbriata</i> sowie Test-Eingabemuster der einzelnen Jahre . .	73
4.6	Vergleich zwischen Messung und Prognose sowie Variabilität der Daten bei der Vorhersage der jährlichen Abundanz von <i>Apatania fimbriata</i> . . .	73
4.7	Möglichkeiten der Visualisierung einer 6×1 -SOM/RBF-Schicht mit Abundanzdaten von <i>Apatania fimbriata</i>	75

4.8	U-Matrix-Darstellung der SOM für Wetter- und Abfluss-Daten	84
4.9	Sammon-Abbildung von SOM-Kodebuchvektoren	85
4.10	Werte der Komponenten der Kodebuchvektoren auf einem SOM-Gitter	86
4.11	Kodebuchvektoren als Linien-Grafiken in der Anordnung der SOM	86
4.12	Kodebuchvektoren (reskaliert) als Balkendiagramme in der Anordnung der SOM	87
4.13	Trajektorie der Siegerneuronen für die zu den 31 Tagen im Dezember 1999 gehörenden Netzeingaben	87
4.14	Trajektorie der SOM-Sieger für die Netzeingabemuster des jeweils ersten Tages eines Monats im Jahr 2000.	87
4.15	SOM-Gitter mit der Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$ von Lerndaten, die in die jeweiligen Einzugsgebiete der SOM-Neuronen fallen	88
4.16	Graustufendarstellung von Likelihoods des Parzen-Dichteschätzers und des VI-Netzes für jeden Kodebuchvektor auf dem SOM-Gitter	88
4.17	Spannweiten $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L) - \min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ der Ausgabegröße und Ausgaben des RBFNs bei der Eingabe von Kodebuchvektoren (Graustufen)	89
4.18	Graustufen-kodierte Konfidenz- und Prognoseintervallbreiten eines VI-Netzes für die Kodebuchvektoren auf dem SOM-Gitter	90
4.19	Messwerte und Prognosen des RBF SOM-Netzwerkes für die Testdaten als Zeitreihendarstellung und Streudiagramm	90
4.20	Histogramm der Residuen für die Prognose mit den Testdaten	91
4.21	Residuen bei der Prognose des täglichen Abflusses für die einzelnen Testdatensätze.	91
4.22	Abhängigkeit der Residuen von den Werten der Prognosen für die Testdaten.	92
4.23	Minimale Abstände der SOM-Kodebuchvektoren zu den Netzeingabevektoren für die Testdaten	92
4.24	Maximale Aktivitäten der RBF-Neuronen über den Nummern der Testdatensätze	93
4.25	Beträge der Residuen über den minimalen Abständen der Testeingabevektoren zu den Kodebuchvektoren	93
4.26	Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$ von Lerndaten-Eingabevektoren, die in dieselben Einzugsgebiete der SOM-Neuronen wie die Test-Eingabedaten fallen	94
4.27	Spannweiten zwischen den Werten der Ausgabegröße in den einzelnen SOM-Klassen	94
4.28	Prognose von Y durch die Mittelwerte der Elemente von $\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L$	95
4.29	Testdaten mit dem durch $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ gegebenen Intervall	95
4.30	Testdaten der abhängigen Größe mit um die Prognosewerte gezeichneten Fehlerbalken (Predictive-Error-Bars)	96
4.31	Anzahl $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^T$ von Testdaten, die in die jeweiligen Einzugsgebiete der SOM-Neuronen fallen	96

4.32	SOM-Gitter mit den für die SOM-Neuronen-Einzugsgebiete zu erwartenden Fehlern (RMSE) und den jeweiligen Anzahlen von Testdaten	97
4.33	Graustufendarstellung auf dem SOM-Gitter relativierter euklidischer Abstände der Kodebuchvektoren und der Test-Eingaben 129 und 272	97
4.34	Aktivitäten der einzelnen RBF-Neuronen des RBFSOM-Netzes auf dem SOM-Gitter für vier ausgewählte Eingaben	98
4.35	Relativierte Aktivitäten der auf das SOM-Gitter abgebildeten RBF-Neuronen-Aktivitäten für zwei ausgewählte Testdaten-Eingabevektoren als Graustufenbild	98
4.36	Ausgabefunktion des RBFSOM-Netzes in Abhängigkeit von zwei veränderlichen (X_1 und X_5) sowie 5 konstant gehaltenen Größen	99
4.37	Streudiagramme der Testdaten für die Netzeingabegrößen X_1 , X_2 und X_3	100
4.38	Partielle Ausgabefunktion des RBFSOM-Netzes mit veränderlicher Niederschlagsmenge am Messpunkt „Schlitz“ in den 7 Tagen vor der Prognose mit maximaler und die mittlerer Aktivität der RBF-Neuronen	100
4.39	Likelihoods (Parzen-Schätzer und VI-Netz) sowie $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$	101
4.40	Abfluss-Prognose in Abhängigkeit von Werten x_1 , erwartete Fehler nach dem Fehlerbalken-Ansatz, $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$	102
4.41	Partielle Ausgabefunktion des RBFSOM-Netzes und maximale RBF-Neuronen-Aktivität bei veränderlicher (an „Haus B“ am Breitenbach während der vier Wochen vor der Prognose gemessener) Lufttemperatur und konstanten restlichen sechs Eingabegrößen für Werte des Testdatensatzes 272	102
4.42	Likelihoods des Parzen- und des VI-Netz-Dichteschätzers sowie der Größe $\#\mathcal{X}_{\kappa(\mathbf{x}(t))}^L$	103
4.43	Abfluss-Prognose in Abhängigkeit von Werten x_5 , erwartete Fehler nach dem Fehlerbalken-Ansatz, $\min(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$ und $\max(\mathcal{Y}_{\kappa(\mathbf{x}(t))}^L)$	103
4.44	Schätzungen der Konfidenz- und Prognoseintervalle (VI-Netz), Abflussprognose und maximale RBF-Neuronen-Aktivität	104
4.45	Trajektorie der SOM-Sieger für sich im gemessenen Wertebereich verändernder, vierwöchiger Lufttemperaturmittel und sonst konstanter Netzeingaben	104
4.46	Wichtigkeit der Prädiktoren, indem jeweils einer auf seinen Mittelwert gesetzt und der RMSE für die Testdaten berechnet wurde.	105
4.47	Messwerte und Prognosen des Breitenbach-Abfluss-Tagesmittels durch ein RBFSOM-Netzwerk mit zusätzlichen linearen Zellen auf der Basis von 20 Prädiktoren, u. a. den Vortagesabflusswerten und ihren Änderungen	106
5.1	Eindimensionale SOM (Kette) zur Beschreibung eines Punktegitters im \mathbb{R}^2	117
6.1	Jährliche Abundanzen von 40 EPT-Arten am Breitenbach	122

6.8 Tabellenverzeichnis

3.1	Betriebsmodi von <i>visualrbfn.oct</i>	39
3.3	Bedeutung des Inhaltes der Datei <i>work_results.txt</i>	58
4.1	Ausgaben von <i>Visualrbfn</i> in der Datei <i>work_results.txt</i> in einer Fallstudie	74
4.2	Selektierte Prädiktoren für ein Wetter-Abfluss-Modell des Breitenbaches .	83
6.1	Artnamen ausgewählter Eintags-, Stein- und Köcherfliegen	121

6.9 Quellenverzeichnis

- [AB54] ANDREWARTHA, H. und BIRCH, L. (1954): *The distribution and abundance of animals*; Chicago: University of Chicaco Press.
- [ABGT00] AMINZADEH, F., BARHEN, J., GLOVER, C. und TOOMARIAN, N. (2000): *Reservoir parameter estimation using a hybrid neural network*; *Computers & Geosciences*; Bd. 26: S. 869–875.
- [And84] ANDĚL, J. (1984): *Statistische Analyse von Zeitreihen*; Berlin: Akademie-Verlag; 1. Aufl.
- [Bel61] BELLMAN, R. (1961): *Adaptive control processes: a guided tour*; Princeton: Princeton University Press.
- [BEPW96] BACKHAUS, K., ERICHSON, B., PLINKE, W. und WEIBER, R. (1996): *Multivariate Analysemethoden: eine anwendungsorientierte Einführung*; Berlin u. a.: Springer; 8. Aufl.
- [BFM96] BRAUN, H., FEULNER, J. und MALAKA, R. (1996): *Praktikum neuronale Netze*; Berlin u. a.: Springer; 1. Aufl.
- [BHT91] BEGON, M., HARPER, J. L. und TOWNSEND, C. R. (1991): *Ökologie: Individuen, Populationen und Lebensgemeinschaften*; Basel u. a.: Birkhäuser; 1. Aufl.
- [Bis94] BISHOP, C. M. (1994): *Novelty detection and neural network validation*; *IEE Proceedings: Vision, Image and Signal Processing*; Bd. 141 (4): S. 217–222.
- [Bis95] BISHOP, C. M. (1995): *Neural networks for pattern recognition*; Oxford: Oxford University Press; 1. Aufl.
- [BLDB96] BARAN, P., LEK, S., DELACOSTE, M. und BELAUD, A. (1996): *Stochastic models that predict trout population density or biomass on a mesohabitat scale*; *Hydrobiologia*; Bd. 337: S. 1–9.
- [Bor93] BORTZ, J. (1993): *Statistik für Sozialwissenschaftler*; Berlin u. a.: Springer; 4. Aufl.
- [Bro94] BROCKHAUS (1994): *Brockhaus-Enzyklopädie in 24 Bänden*; Mannheim: F. A. Brockhaus; 19. Aufl.
- [BSW⁺97] BORCHARDT, D., SCHLEITER, I., WERNER, H., DAPPER, T. und SCHMIDT, K.-D. (1997): *Modellierung ökologischer Zusammenhänge in Fließgewässern mit Neuronalen Netzwerken*; *Wasser und Boden*; Bd. 49: S. 38, 47–50.

- [CBV01] COTTRELL, M., DE BODT, E. und VERLEYSSEN, M. (2001): *A statistical tool to assess the reliability of self-organizing maps*; in: *Advances in Self-Organising Maps*; S. 7–14; Lincoln: Springer.
- [CCB99] CARNEY, J., CUNNINGHAM, P. und BHAGWAN, U. (1999): *Confidence and prediction intervals for neural network ensembles*.
<http://citeseer.nj.nec.com/carney99confidence.html>
- [CD98] CHINNAM, R. B. und DING, J. (1998): *Prediction limit estimation for neural network models*; *IEEE Transactions on Neural Networks*; Bd. 9 (6): S. 1515–1522.
- [Cha93] CHATFIELD, C. (1993): *Neural networks: forecasting breakthrough or just passing fad?*; *International Journal of Forecasting*; Bd. 9: S. 1–3.
- [CI00] COHEN, S. und INTRATOR, N. (2000): *A hybrid projection based and radial basis function architecture*; in: *Multiple Classifier Systems*; S. 147–156.
<http://citeseer.nj.nec.com/cohen00hybrid.html>
- [CLR96] CHRYSOLOURIS, G., LEE, M. und RAMSEY, A. (1996): *Confidence interval prediction for neural network models*; *IEEE Transactions on Neural Networks*; Bd. 7 (1): S. 229–232.
- [Cox90] COX, E. J. (1990): *Studies on the algae of a small softwater stream; occurrence and distribution with particular reference to the diatoms*; in: *Archiv für Hydrobiologie/Supplementband*; Bd. 83; S. 525–552.
- [CPMC96] CHON, T.-S., PARK, Y., MOON, K. und CHA, E. (1996): *Patternizing communities by using an artificial neural network*; *Ecological Modelling*; Bd. 90: S. 69–78.
- [Dap98] DAPPER, T. (1998): *Dimensionsreduzierende Vorverarbeitungen für Neuronale Netze mit Anwendungen in der Umweltmodellierung*; Aachen: Shaker.
- [DR01] DYBOWSKI, R. und ROBERTS, S. J. (2001): *Confidence intervals and prediction intervals for feed-forward neural networks*; in: DYBOWSKI, R. und GANT, V. (Hg.), *Clinical Applications of Artificial Neural Networks*; S. 298–326; Cambridge: Cambridge University Press.
<http://www.dybowski.com/papers/nnerr.pdf>
- [Eat97] EATON, J. W. (1997): *GNU Octave — a high-level interactive language for numerical computations*; 3. Aufl.; octave.dvi; Octave-Version: 2.0.13.
<http://www.octave.org/>
- [Eat02] EATON, J. W. u. a.. (2002): *Octave (Homepage)*.
<http://www.octave.org/>

- [ET93] EFRON, B. und TIBSHIRANI, R. B. (1993): *An introduction to the bootstrap*; London: Chapman and Hall.
- [Ewe90] EWERT, J.-P. (1990): *Nerven- und Sinnesphysiologie*; Braunschweig: Westermann; 1. Aufl.
- [FJ98] FERNANDO, D. und JAYAWARDENA, A. (1998): *Runoff forecasting using RBF networks with OLS algorithm*; *Journal of Hydrologic Engineering*; Bd. 3 (8): S. 203–209.
- [Foo99] FOODY, G. (1999): *Applications of the self-organising feature map neural network in community data analysis*; *Ecological Modelling*; Bd. 120: S. 97–107.
- [Fre94] FREEMAN, J. A. (1994): *Simulating neural networks with Mathematica*; Reading (MA) u. a.: Addison-Wesley; 1. Aufl.
- [Fri94] FRITZKE, B. (1994): *Fast learning with incremental RBF networks*; *Neural Processing Letters*; Bd. 1 (1): S. 2–5.
<http://citeseer.nj.nec.com/fritzke94fast.html>
- [GKN78] GELLERT, W., KÄSTNER, H. und NEUBER, S. (Hg.) (1978): *Fachlexikon ABC Mathematik*; Thun u. a.: Verlag Harri Deutsch; 1. Aufl.
- [Gol89] GOLDBERG, D. E. (1989): *Genetic algorithms in search, optimization & machine learning*; Reading (MA) u. a.: Addison-Wesley; korr. Nachdruck d. 1. Aufl.
- [Grz93] GRZIMEK, B. (Hg.) (1993): *Grzimeks Tierleben: Enzyklopädie des Tierreichs in 13 Bänden*; Bd. 2 (Insekten); München: Deutscher Taschenbuch Verlag; unveränd. Nachdruck d. dtv-Ausgabe 1979/80.
- [Gt92] GILLY, D. und THE STAFF OF O'REILLY & ASSOCIATES (1992): *UNIX in a nutshell: desktop quick reference for SV & Solaris 2.0*; O'Reilly & Associates; 2. Aufl.
- [HDK00] HOEKSTRA, A., DUIN, R. P. und KRAAIJVELD, M. A. (28.02.2000): *Neural networks applied to data analysis*.
<http://valhalla.ph.tn.tudelft.nl/generalization/papers/leondes/node10.html>
- [Hes97] HESKES, T. (1997): *Practical confidence and prediction intervals*; in: MOZER, M. C., JORDAN, M. I. und PETSCHKE, T. (Hg.), *Advances in Neural Information Processing Systems*; Bd. 9; S. 176; The MIT Press.
<http://citeseer.nj.nec.com/heskes97practical.html>

- [Hof93] HOFFMANN, N. (1993): *Kleines Handbuch Neuronale Netze: anwendungsorientiertes Wissen zum Lernen und Nachschlagen*; Braunschweig u. a.: Vieweg; 1. Aufl.
- [HZB02] HSIEH, B. B., ZHANG, B. und BARTOS, C. L. (2002): *Use of artificial neural networks in a streamflow prediction system*.
<http://www.nd.com/public/appsum/app-predict.doc>
- [Ill71] ILLIES, J. (1971): *Emergenz 1969 im Breitenbach; Archiv für Hydrobiologie*; Bd. 69: S. 14–59.
- [JF98] JAYAWARDENA, A. und FERNANDO, D. (1998): *Use of radial basis function type artificial neural networks for runoff simulation; Computer-Aided Civil and Infrastructure Engineering*; Bd. 13 (2): S. 91–99.
- [KHKL96] KOHONEN, T., HYNINEN, J., KANGAS, J. und LAAKSONEN, J. (1996): *SOM_PAK — the self-organizing map program package*; Techn. Ber. A31; Helsinki University of Technology, Laboratory of Computer and Information Science; Espoo (Finnland); Version 3.1.
http://cochlea.hut.fi/research/papers/som_tr96.ps.Z
- [Klö94] KLÖPPEL, B. (1994): *Stabilität und Kapazität Neuronaler Netzwerke am Beispiel der EEG-Analyse*; Aachen: Shaker; 1. Aufl.
- [KLTP99] KINDERMANN, L., LEWANDOWSKI, A., TAGSCHERER, M. und PROTZEL, P. (1999): *Computing confidence measures and marking unreliable predictions by estimating input data densities with MLPs*; in: *Proceedings of the Sixth International Conference on Neural Information Processing (ICONIP '99), Perth*; S. 91–94.
- [Koh82] KOHONEN, T. (1982): *Self-organized formation of topologically correct feature maps*; *Biological Cybernetics*; Bd. 43: S. 59–69.
- [Koh01] KOHONEN, T. (2001): *Self-organizing maps*; Berlin u. a.: Springer; 3. Aufl.
- [Kre68] KREYSZIG, E. (1968): *Statistische Methoden und ihre Anwendungen*; Göttingen: Vandenhoeck & Ruprecht; 3. Aufl.
- [KSV92] KÖHLER, W., SCHACHTEL, G. und VOLESKE, P. (1992): *Biostatistik: Einführung in die Biometrie für Biologen und Agrarwissenschaftler*; Berlin u. a.: Springer; 1. Aufl.
- [KVHK97] KOSKELA, T., VARSTA, M., HEIKKONEN, J. und KASKI, K. (1997): *Time series prediction using recurrent SOM with local linear models*; Techn. Ber.

- B15; Helsinki University of Technology, Laboratory of Computational Engineering; Espoo (Finland).
<http://citeseer.nj.nec.com/koskela97time.html>
- [LBB⁺96] LEK, S., BELAUD, A., BARAN, P., DIMOPOULOS, I. und DELACOSTE, M. (1996): *Role of some environmental variables in trout abundance models using neural networks*; *Aquatic Living Ressources*; Bd. 9: S. 23–29.
- [LDDEG96] LEK, S., DIMOPOULOS, I., DERRAZ, M. und EL GHACHTOUL, Y. (1996): *Modélisation de la relation pluie-débit à l'aide des réseaux de neurones artificiels*; *Revue des sciences de l'eau*; Bd. 3: S. 319–331.
- [LKU92a] LEONARD, J. A., KRAMER, M. A. und UNGAR, L. H. (1992): *A neural network architecture that computes its own reliability*; *Computers & Chemical Engineering*; Bd. 16 (9): S. 819–835.
- [LKU92b] LEONARD, J. A., KRAMER, M. A. und UNGAR, L. H. (1992): *Using radial basis functions to approximate a function and its error bounds*; *IEEE Transactions on Neural Networks*; Bd. 3 (4): S. 624–626.
- [Loh01] LOHNINGER, H. (2001): *Teach/Me – Datenanalyse*; Berlin: Springer; Einzelplatz-Version (CD-ROM/Windows).
- [LS95] LORRAI, M. und SECHI, G. M. (1995): *Neural nets for modelling rainfall-runoff transformations*; *Water Resources Management*; Bd. 9 (4): S. 299–213.
- [LZ98] LOWE, D. und ZAPART, K. (1998): *Point-wise confidence interval estimation by neural networks: a comparative study based on automotive engine calibration*; Techn. Ber.; Neural Computing Research Group, Dept. of Computer Science & Applied Mathematics, Aston University, Birmingham (UK).
<http://www.ncrg.aston.ac.uk/>
- [Mac67] MACQUEEN, J. (1967): *Some methods for classification and analysis of multivariate observations*; in: LECAM, L. M. und NEYMAN, J. (Hg.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*; Bd. I; S. 281–297; Berkeley: University of California Press.
- [Mac92] MACKAY, D. (1992): *Bayesian interpolation*; *Neural Computation*; Bd. 4: S. 415–447.
- [Mac95] MACKAY, D. (1995): *Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks*.
<ftp://wol.ra.phy.cam.ac.uk/pub/www/mackay/network.ps.gz>

- [Mal02] MALICKY, H. (2002): *A quantitative field comparison of different types of emergence traps in a stream: general, Trichoptera, Diptera (Limoniidae and Empididae)*; *Annales de Limnologie – International Journal of Limnology*; S. 133–149.
- [Mas93] MASTERS, T. (1993): *Practical neural network recipes in C++*; London: Academic Press, Inc.; 1. Aufl.
- [MD96] MAIER, H. R. und DANDY, G. C. (1996): *The use of artificial neural networks for the prediction of water quality parameters*; *Water Resources Research*; Bd. 32 (4): S. 1013–1022.
- [MD00] MAIER, H. R. und DANDY, G. C. (2000): *Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications*; *Environmental Modelling & Software*; Bd. 15: S. 101–124.
- [Mic92] MICHELS, P. (1992): *Nichtparametrische Analyse und Prognose von Zeitreihen*; Heidelberg: Physica-Verlag; 1. Aufl.
- [MLD97] MASTRORILLO, S., LEK, S. und DAUBA, F. (1997): *Predicting the abundance of minnow *Phoxinus phoxinus* (Cyprinidae) in the River Ariège (France) using artificial neural networks*; *Aquatic Living Resources*; Bd. 10: S. 169–176.
- [MZ99] MARTEN, M. und ZWICK, P. (1999): *3.10 Emergenz*; in: TÜMPLING, W. v. und FRIEDRICH, G. (Hg.), *Methoden der Biologischen Gewässeruntersuchung 2*; S. 227–238; Jena u. a.: Fischer.
- [Nea93] NEAL, R. (1993): *Bayesian learning via stochastic dynamics*; in: GILES, C., HANSON, S. und COWAN, J. (Hg.), *Advances in Neural Information Processing Systems*; Bd. 5; S. 475–482; San Mateo (CA): Morgan Kaufmann.
- [Nea96] NEAL, R. (1996): *Bayesian learning for neural networks*; New York: Springer; 1. Aufl.
- [NKK94] NAUCK, D., KLAWONN, F. und KRUSE, R. (1994): *Neuronale Netze und Fuzzy-Systeme*; Vieweg: Wiesbaden; 1. Aufl.
- [NW95] NIX, D. A. und WEIGEND, A. S. (1995): *Learning local error bars for non-linear regression*; in: TESAURO, G., TOURETZKY, D. S. und LEEN, T. K. (Hg.), *Advances in Neural Information Processing Systems 7 (NIPS '94)*; S. 489–496; Cambridge (MA): MIT Press.

- [Oba98] OBACH, M. (1998): *Anwendung statistischer Methoden und künstlicher Neuronaler Netzwerke im Vergleich*; Diplomarbeit; Fachbereich Mathematik/Informatik der Universität Kassel; Kassel.
<http://www.Michael-Obach.de/>
- [OWWS01] OBACH, M., WAGNER, R., WERNER, H. und SCHMIDT, H.-H. (2001): *Modelling population dynamics of aquatic insects with artificial neural networks*; *Ecological Modelling*; Bd. 146 (1-3): S. 207–217.
- [PEM00] PAPADOPOULOS, G., EDWARDS, P. J. und MURRAY, A. F. (2000): *Confidence estimation methods for neural networks: a practical comparison*.
http://www.ee.ed.ac.uk/~neural/research/papermaking/papers/gp_ESANN00.pdf
- [PG90] POGGIO, T. und GIROSI, F. (1990): *Networks for approximation and learning*; in: *Proceedings of the IEEE*; Bd. 78; S. 1481–1497.
- [PKTL00] PROTZEL, P., KINDERMANN, L., TAGSCHERER, M. und LEWANDOWSKI, A. (2000): *Abschätzung der Vertrauenswürdigkeit von Neuronalen Netzprognosen bei der Prozessoptimierung*; in: *VDI-Bericht 1626 zur Fachtagung der VDI/VDE Gesellschaft für Mess- und Automatisierungstechnik, Baden-Baden*; S. 335–339.
- [Pla93] PLATE, E. J. (1993): *Statistik und angewandte Wahrscheinlichkeitslehre für Bauingenieure*; Berlin: Ernst & Sohn; 1. Aufl.
- [Pof92] POFF, N. (1992): *Why disturbance can be predictable: a perspective on the definition of disturbance in streams*; *Journal of the North American Benthological Society*; S. 86–92.
- [PR97] PENNY, W. und ROBERTS, S. (1997): *Neural network predictions with error bars*.
<http://citeseer.nj.nec.com/penny97neural.html>
- [PR98] PENNY, W. D. und ROBERTS, S. J. (1998): *Error bars for linear and non-linear neural network regression models*.
<http://citeseer.nj.nec.com/penny98error.html>
- [PTVF95] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T. und FLANNERY, B. P. (1995): *Numerical recipes in C: the art of scientific computing*; Cambridge u. a.: Cambridge University Press; 2. Aufl.
- [Rem92] REMMERT, H. (1992): *Ökologie: ein Lehrbuch*; Berlin u. a.: Springer; 5. Aufl.

- [RHST94] RESH, V. H., HILDREW, A. G., STATZNER, B. und TOWNSEND, C. R. (1994): *Theoretical habitat templates, species traits, and species richness: a synthesis of long-term research on the upper Rhône river in the context of concurrently developed ecological theory*; *Freshwater Biology*; Bd. 31: S. 539–554.
- [Rin74] RINGE, F. (1974): *Die Chironomiden-Emergenz 1970 in Breitenbach und Rohrwiesenbach*; *Archiv für Hydrobiologie/Supplementband*; Bd. 45: S. 212–304.
- [RMS90] RITTER, H., MARTINETZ, T. und SCHULTEN, K. (1990): *Neuronale Netze: Eine Einführung in die Neuroinformatik selbstorganisierter Netzwerke*; Bonn u. a.: Addison-Wesley; 1. Aufl.
- [Roj93] ROJAS, R. (1993): *Theorie der neuronalen Netze: Eine systematische Einführung*; Berlin u. a.: Springer.
- [Ros62] ROSENBLATT, F. (1962): *Principles of neurodynamics*; New York: Spartan.
- [RPP97] ROBERTS, S. J., PENNY, W. und PILLOT, D. (1997): *Novelty, confidence & errors in connectionist systems*.
<http://www.fil.ion.ucl.ac.uk/~wpenny/publications/ieefault.ps>
- [Sam69] SAMMON, J. (1969): *A nonlinear mapping for data structure analysis*; *IEEE Transactions on Computers*; Bd. C-185: S. 401–409.
- [Sar02] SARLE, W. S. (09.06.2002): *Periodic Posting to the Usenet newsgroup comp.ai.neural-nets*.
<ftp://ftp.sas.com/pub/neural/FAQ.html>
- [SBW⁺99] SCHLEITER, I., BORCHARDT, D., WAGNER, R., DAPPER, T., SCHMIDT, K.-D., SCHMIDT, H.-H. und WERNER, H. (1999): *Modelling water quality, bioindication and population dynamics in lotic ecosystems using neural networks*; *Ecological Modelling*; Bd. 120: S. 271–286.
- [Sch93] SCHÖNEBURG, E. H. (1993): *Industrielle Anwendung Neuronaler Netze*; Bonn u. a.: Addison-Wesley; 1. Aufl.
- [Sch97] SCHERER, A. (1997): *Neuronale Netze — Grundlagen und Anwendungen*; Braunschweig u. a.: Vieweg.
- [SH92] SCHIØLER, H. und HARTMANN, U. (1992): *Mapping neural network derived from the Parzen window estimator*; *Neural Networks*; Bd. 5: S. 903–909.

- [SMZM97] SHAO, R., MARTIN, E. B., ZHANG, J. und MORRIS, A. J. (1997): *Confidence bounds for neural network representations*; *Computers & Chemical Engineering*; Bd. 21 (Suppl. 1): S. S1173–S1178.
- [Spe91] SPECHT, D. F. (1991): *A general regression network*; *IEEE Transactions on Neural Networks*; Bd. 2 (6): S. 568–576.
- [SS97] SCHLITTEGEN, R. und STREITBERG, B. H. J. (1997): *Zeitreihenanalyse*; München u. a.: Oldenbourg; 7. Aufl.
- [ST83] SCHAEFER, M. und TISCHLER, W. (1983): *Ökologie*; Stuttgart u. a.: Fischer; 2. Aufl.
- [TKLP99] TAGSCHERER, M., KINDERMANN, L., LEWANDOWSKI, A. und PROTZEL, P. (1999): *Overcome neural limitations for real world applications by providing confidence values for network predictions*; in: *Proceedings of the Sixth International Conference on Neural Information Processing (ICONIP '99), Perth*; S. 520–525.
- [Tra01] TRAJAN SOFTWARE LTD. (2001): *Trajan 6.0 Neural Network Simulator*; Durham; Dokument-Nr. TR/UM/60.1.
- [Ult92] ULTSCH, A. (1992): *Self-organizing neural networks for visualization and classification*.
<http://www.Mathematik.Uni-Marburg.de/~wina/PapersHtml/ultsch92b/ultsch92b.html>
- [Ult93] ULTSCH, A. (1993): *Self-organized feature maps for monitoring and knowledge acquisition of a chemical process*; in: GIELEN, S. und KAPPEN, B. (Hg.), *Proceedings of the International Conference on Artificial Neural Networks ICANN 93*; S. 864–867; London: Springer.
<http://www.Mathematik.Uni-Marburg.de/~nkisec/Papers/ICANNamsterdam.html>
- [Ves97] VESANTO, J. (1997): *Using the SOM and local models in time-series prediction*; in: *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4–6*; S. 209–214; Espoo (Finnland): Helsinki University of Technology, Neural Networks Research Centre.
<http://citeseer.nj.nec.com/vesanto97using.html>
- [Voß00] VOSS, W. (2000): *Taschenbuch der Statistik*; München u. a.: Fachbuchverlag Leipzig im Hanser-Verlag; 1. Aufl.
- [Wah86] WAHRIG, G. H. (1986): *Deutsches Wörterbuch*; Gütersloh u. a.: Bertelsmann Lexikon Verlag; Jubiläumsausgabe.

- [WDS00] WAGNER, R., DAPPER, T. und SCHMIDT, H.-H. (2000): *The influence of environmental variables on the abundance of aquatic insects — a comparison of ordination and artificial neural networks*; *Hydrobiologia*; Bd. 422/423: S. 143–152.
- [WN94] WEIGEND, A. und NIX, D. (1994): *Predictions with confidence intervals (local error bars)*; Seoul (Korea).
<http://citeseer.nj.nec.com/weigend94predictions.html>
- [Wol99] WOLFRAM RESEARCH (1999): *Mathematica 4.0 standard add-on packages*; Champaign (IL): Wolfram Media.
- [WSM94] WAGNER, R., SCHMIDT, H.-H. und MARXSEN, J. (1994): *The hyporheic habitat of the Breitenbach, spatial structure and physicochemical conditions as a basis for benthic life*; *Limnologica*; Bd. 23: S. 285–294.
- [YKC⁺00] YANG, L., KAVLI, T., CARLIN, M., CLAUSEN, S. und DE GROOT, P. F. M. (2000): *An evaluation of confidence bound estimation methods for neural networks*.
- [Zah89] ZAHRADNIK, J. (1989): *Der Kosmos-Insektenführer: ein Bestimmungsbuch*; Stuttgart: Franckh; 5. Aufl.
- [ZBS99] ZEALAND, C. M., BURN, D. H. und SIMONOVIC, S. P. (1999): *Short term streamflow forecasting using artificial neural networks*; *Journal of Hydrology*; Bd. 214: S. 32–48.
- [Zel94] ZELL, A. (1994): *Simulation Neuronaler Netze*; Bonn u. a.: Addison-Wesley; 1. Aufl.
- [ZFH94] ZHU, M., FUJITA, M. und HASHIMOTO, N. (1994): *Application of neural networks to runoff prediction*; *Stochastic and Statistical Methods in Hydrology and Environmental Engineering*; Bd. 3: S. 205–216.
- [Zwi93] ZWICK, P. (1993): *Fließgewässergefährdung durch Insektizide*; *Naturwissenschaften*; Bd. 79: S. 437–442.

6.10 Stichwortverzeichnis

-

-buffer, 63
 -case, 63
 -cin, 63
 -cnuminfname, 63
 -din, 63
 -epsbasename, 63
 -framed, 63
 -help, 63
 -labels, 63
 -ndef, 63
 -ndefstr, 63
 -negative, 63
 -ps, 64
 -squares, 64

A

a-posteriori-Wahrscheinlichkeitsdichte, 22
 a-priori-Wahrscheinlichkeitsdichte, 22
 Abbildung, 11
 Abfluss, 69, 71–77, 79–110
 abhängige Variable, 2
 Abhängigkeit
 stochastische, 2, 77
 absolutes Glied, 47
 Abstand
 euklidischer, 11, 17
 abundance, → Abundanz
 Abundanz, 67
 Adaptionsschritt (SOM-Training), 13
 affine Funktion, 33
 Aktivität (eines Neurons), 9, 17
 Algorithmus
 Genetischer, 36, 69, 70
Amphinemura standfussi, 71
 Anwendungsgebiete
 von KNNs, 9
Apatania fimbriata, 69, 72–77
 Arbeitsmodus, 39
 ASCII, 37
 Assoziator
 linearer, 26, 114
 Aufgabenbereiche
 von KNNs, 9
 Ausdehnung (einer Basisfunktion), 17

Ausgabe, 2
 Ausgabedatenraum, → Ausgaberaum
 Ausgaberaum, 2
 Ausgabeschicht, 10
 Ausreißer, 36
 Außenpunkt, 36
Awk (Software), 36, 57, 60, 85

B

Bachforelle, 68
 backpropagation of errors, → Fehlerrückführung
 Bandbreite, 17
 Basisfunktion
 elliptische, 30
 gaußsche, 17
 lokale, 17
 nicht-lokale, 17
 radiale, 17
 Batch-Datei, 35
 Batch-Verfahren, 31
 Bayesian network, → bayesisches Netzwerk
 Bayesianisches Netzwerk, → bayesisches Netzwerk
 bayesisches Netzwerk, 22
 Bereichsschätzung, 4
 Bestimmtheitsmaß, 3, 57, 90, 105, 106, 112
 Betriebsmodus, 38
 Bias (eines Modells), 24
 Bias-Element, 17, 20, 26, 33, 47
 Bioindikation, 77
 Biotop, 67
 Biozönose, 77, 107, → Lebensgemeinschaft
 Black-box, 115
 Blasenfuß, 68
 bool, 41
 Bootstrap-Komitee, 24
 Bootstrap-Verfahren, 24, 111
 Box-Whisker-Plots, 73, 113
 BPContinue, 44
 BPetaS, 44
 BPetaW, 44
 BPetaZ, 44
 BPLogStepNumber, 44
 BPMomentumB, 44
 BPNumSteps, 44
 BPStopRMSE, 44
 BPSVD, 44

Breite (einer Basisfunktion), 17
Breitenbach, 68, 81

C

call-by-reference, → Referenzaufruf
Ceteris-paribus-Prinzip, 78
City-Block-Metrik, 17
Clusteranalyse, 9, 11, 18
committee, → Komitee
community, → Lebensgemeinschaft
component plane, → Komponentenebene
confidence interval, → Konfidenzintervall
Counterpropagation-Netzwerk, 26, 114
cross-validation, → Kreuzvalidierung
curse of dimensionality, → Fluch der Dimensionen
CVCrossValidate, 44
CVNumValsamples, 44
CVRandomOrder, 45
CVRatioValsamples, 45
CVReuse, 45
CVShowProgress, 45
Cygwin (Software), 35, 38, 58–60, 62, 65, 128

D

DataFileName, 45
DataInputColumns, 45
DataOutputColumns, 45
DataRescaleInput, 45
Datenreduktion, 42, 48
Delta-Methode, 24
determination coefficient, → Bestimmtheitsmaß
deviation, → Ausdehnung (einer Basisfunktion)
Dimensionsreduktion, 9
discharge, → Abfluss
Diskriminanzanalyse, 9
Dispersionsmaß, 6, 111
Distorsionsmaß, mittleres, 113
Disturbance-Konzept, 78
Drift, 77

E

Eingabe, 2
Eingabedatenraum, → Eingaberaum
Eingaberaum, 2
Eingabeschicht, 10
Einschrittprognose, 112
Eintags-, Stein- und Köcherfliegen, 69
Eintagsfliege, 69, 121

Einzelfehler, 2
Einzugsgebiet
 eines Fließgewässers, 67
 eines SOM-Neurons, 11
elliptische Basisfunktion, 30
ELMAN-Netz, 110
Elritze, 68
Emergenz-Methode, 79
Emergenzfalle, 69–72, 119
Encapsulated PostScript, 51
Ensemble (von KNNs), 25, 109
enum, 41
Ephemeroptera, 69, 121
EPS, → Encapsulated PostScript
EPT, → Eintags-, Stein- und Köcherfliegen
erklärende Variable, 2
Erregung, → Aktivität
Erregungszentrum, 11, 12
error bars, → Fehlerbalken
euklidischer Abstand, 11, 17
Evapotranspiration, 108
Evolution, 67
Excel (Software), 36, 57, 65
exponential smoothing, → exponentielles Glätten
exponentielles Glätten, 112
Extrapolation, 4
Extrapolationsdetektor, 6, 23

F

Faktorenanalyse, 9, 30, → Hauptkomponentenanalyse
 lyse
feed-forward network, → vorwärts gerichtetes Netzwerk
fehlender Messwert, 36, 45, 62, 109
Fehlerbalken, 20, 28, 29, 47, 56, 89, 95, 115, 116
Fehlerquadratsumme, 3, 10, 18
Fehlerrückführung, 30
Fehlerrückführungsmodus, 39
Feld
 rezeptives, 11
Flaschenhalsnetze, 30
float, 40
flow, → Abfluss
Fluch der Dimensionen, 30, 118
Free Software Foundation, 59
function file, 61
Funktion
 affine, 33
 sinusoide, 32

G

Gaussian basis function, → gaußsche Basisfunktion
 gaußsche Basisfunktion, 17
 Gauß-Funktion, 13
 General Public License, 59, 61, 65
 General Regression Neural Network, 19, 69
 Generalisierungsfähigkeit, 3, 111
 Genetischer Algorithmus, 36, 69, 70
 Gesamtschrittverfahren, 31
 Gewicht
 eines KNNs, 9
 Gewichtsvektor, 11
 gewöhnliches RBFN, 17, 32
 Giftunfall, 78
 Gitter, 11
 Glätten
 exponentielles, 112
 Glättungsparameter, 17
 gleitendes Zeitfenster, 68
 Glied
 absolutes, 47
 globale Prognosequalität, 111
 globales Minimum, 33
Gnuplot (Software), 55, 59, 60, 62, 64, 65, 128
 Gradientenabstiegsverfahren, 18, 30, 39, 42, 44
 grafische Benutzerschnittstelle, 66
 graphical user interface, → grafische Benutzer-
 schnittstelle
 grid, → Gitter
 GRNN, → General Regression Neural Network
GRNN (Software), 36, 118, 128
 Gruppierung, 9
 GUI, → grafische Benutzerschnittstelle

H

Hardwarevoraussetzung, 59
 Hauptkomponentenanalyse, 26, 30, 116, → Fakto-
 renanalyse
 Hebeleffekt, 109
 Hesse-Matrix, 25
 Hessian matrix, → Hesse-Matrix
 Heteroskedastizität, 29
 hidden layer, → verdeckte Schicht
 high-level language, → Hochsprache
 Histogramm, 28
 Hochsprache, 61
 Hold-Out-Validierung, 3, 111
 Homoskedastizität, 29, 91
 Hybridisierung, 5, 26

Hyper-Basisfunktionen-Netz, 30
 Hyperquader, 112, 114

I

input, → Eingabe
 input layer, → Eingabeschicht
 input noise, 24
 integer, 40, 62
 Interaktion, 77
 Interquartilbereich, 75
 Intervallschätzung, 4

J

Jacobi-Matrix, 23
 JORDAN-Netz, 110

K

K-Means-Algorithmus, 18, 20, 43, 114
 Kalibrationsproblem, 2
 Karte
 motorische, 26, 112
 selbstorganisierende, 11
 sensorische, 11
 Kern, 19
 Kette, 11, 43
 KFM, → selbstorganisierende Karte
 Klassifikation, 9
 Kleinstquadrat-Schätzung, 18, 20
 KNN, → Künstliches Neuronales Netzwerk
 Köcherfliege, 69, 77, 121
 Kodebuchvektor, 11
 Kodebuchvektorkomponente, 16, 56
 KOHONEN, 11
 Kohonen topographic feature map, → Kohonens to-
 pographische Merkmalskarte
 Kohonen's feature map, → selbstorganisierende
 Karte
 Kohonens topographische Merkmalskarte, 11
 Komitee, 24
 Komponente
 eines Kodebuchvektors, 16, 56
 saisonale, 112
 Komponentenebene, 16
 Konfidenzintervall, 4
 Kontextzelle, 110
 konzeptionelles Modell
 zur Abflussprognose, 80
 Korrelationsanalyse, 77

Korrelationskoeffizient, 92, 93, 109
 Kreuzvalidierung, 3, 21, 42, 111
 Kreuzvalidierungsmodus, 39
 Künstliches Neuronales Netzwerk, 1, 9

L

lattice, → Gitter
 Laufzeit von *visualrbfn.oct*, 51, 57–58, 65
 layer, → Schicht (eines KNNs)
 learning, → Lernen
 Learning Vector-Quantization, → Vektorquantisierung
 Leave-one-out-Methode, 4
 Lebensgemeinschaft, 77, → Biozönose
 Lerndaten, 3
 für Abflussprognosen, 82
 Lernen, 3, 10
 einer SOM, 12
 eines RBFNs, 17
 mit Lehrer, 10
 ohne Lehrer, 10
 Lernende Vektorquantisierung, 18
 Lernmodus, 39
 Lernrate, 13
 Levenberg-Marquardt-Verfahren, 32, 119
 Likelihood, 22, 23, 43, 46, 88
 Limnologie, 67
 linear least-squares, → lineare Kleinstquadrate
 lineare Kleinstquadrate, 18
 linearer Assoziator, 26, 114
 Lineares Neuronales Netzwerk, 10, 18
 Link
 symbolischer, 38, 61
 LNN, → Lineares Neuronales Netzwerk
 lokale radiale Basisfunktion, 17
 lokales Minimum, 24, 33
 LVQ, → Lernende Vektorquantisierung

M

mapping, → Abbildung
 Markov-Prozess, 14
Mathematica (Software), 41, 57, 65, 116, 128
Matlab (Software), 57, 61, 128
 Maximum-Likelihood-Schätzer, 22
 Median, 6, 74
 Messwert
 fehlender, 36, 45, 62, 109
 Methode
 schrittweise, 36

Minimum
 globales, 33
 lokales, 24, 33
 missing value, → fehlender Messwert
 Mississippi, 110
 Mittelwert
 einer Klasse, 6
 mittleres Distorsionsmaß, 113
 MLP, → Multi-Layer-Perzeptron
 Mode, 45
 Modell-Bias, 24
 Momentum-Term, 31
 motorische Karte, 26, 112
 MS-DOS-Batch-Datei, 35
 Multi-Layer-Perzeptron, 10, 17, 108
 multivariate Zeitreihe, 81
 Muster, 6, 12, 14, 28, 37, 63, 70
 Musterdatei, 37

N

N^2E^4 (Software), 64, 66, 128
 nachbarschaftserhaltend, 11
 Nachbarschaftsfunktion, 13
 Nachbarschaftsradius, 13
 Nächste-Nachbarn-Heuristik, 19, 20, 46, 105, 114
 Nächster-Nachbar-Klassifizierung, 11, 113
 Nadaraya-Watson-Kern-Regressionsschätzer, 19
 naive Prognose, 82, 105, 107, 109, 112
 NDEF, 45
 Negativ (einer Graustufendarstellung), 97
 neighbourhood kernel, → Nachbarschaftsfunktion
 Nervensystem, 9
 Nervenzelle, 9
 Netzausgabe, 2
 partielle, 99
 Netzeingabe, 2
 Neuartigkeit, 22
 Neuron, 9
 nicht-lokale radiale Basisfunktion, 17
 Nische, 67
 fundamentale, 67
 realisierte, 67, 76
 Normierung, → Skalierung
 novelty, → Neuartigkeit

O

Octave (Software), 21, 35, 37, 38, 40, 41, 57, 60–62, 64, 128, 147
Octave-Skript, 61
 Offline-Verfahren, 31
 one step ahead prediction, → Einschrittprognose
 Optimierungsmodus, 39
options.m, 40, 41, 83
optionscheck.m, 41, 55, 123
optionscheck.txt, 41, 55
 Ordinierung, 78
 output, → Ausgabe
 output layer, → Ausgabeschicht
 overfitting, → Überanpassung

P

Parameterdatei, 38, 40
 partiell rekurrentes Netzwerk, 110
 partielle Netzausgabe, 99
 partitionierendes Verfahren, 18
 Parzen-Dichteschätzer, 22, 43, 46
ParzenBandWidth, 46
ParzenDataFile, 46
ParzenDensity, 46
 pattern, → Muster
 PCA, → Hauptkomponentenanalyse
 Peano-Kurve, 117
 Perceptron Radial Basis Net, 26
 Persistenzmodell, 82
 Perzentil, 75
 Perzeptron, 26
 Perzeptron-Zellen, 26
 Phasendurchschnitt, 112
Phoxinus phoxinus, 68
 Plecoptera, 69, 121
PlotActivities, 46
PlotGraphics, 46
Ploticus (Software), 60, 95, 128
 pointer, → Zeiger
 Populationsdichte, 67
 posterior probability density, → a-posteriori-Wahrscheinlichkeitsdichte
 PostScript, 51
 Prädiktor, 2
 PRBFN, → Perceptron Radial Basis Net
 prediction, → Prognose
 prediction interval, → Prognoseintervall
 Predictive-Error-Bars, 20, 29, 56, 58, 95, 96, 115

principal components analysis, → Hauptkomponentenanalyse
 prior probability density, → a-priori-Wahrscheinlichkeitsdichte
 Prognose, 1
 naive, 82, 105, 107, 109, 112
 Prognose-Variable, 2
 Prognoseintervall, 4
 Prognosequalität
 globale, 111
 Prototyp, 5, 11, 17, 27
 Pruning, 33
 Pseudo-Inverse, 18

Q

Quadratsummenfehler, → Fehlerquadratsumme
 Quantil, 6
 Quantisierungsfehler, 15, 70, 92, 113
 Quartil, 75
 Quiet, 46

R

radial basis function, → radiale Basisfunktion
 radiale Basisfunktion, 16, 17
 Radiale-Basisfunktionen-Netzwerk, 16
 gewöhnliches, 17, 32
 range, 40
 raw data, → Rohdaten
 RBF, → radiale Basisfunktion
 RBF-Netz, → Radiale-Basisfunktionen-Netzwerk
RBFBandWidthConst, 46
RBFBandWidthDefault, 46
RBFBandWidthNN, 46
RBFBandWidthStretch, 47
RBFCalcLearnError, 47
RBFCentresNum, 47
RBFCentresPos, 47
 RBFN, → Radiale-Basisfunktionen-Netzwerk
RBFPESupport, 47
 RBFSOM, 26
RBFType, 47
RBFVISupport, 47
RBFWithBias, 47
RBFWithLinearTerm, 47
 realisierte Nische, 76
ReduceData, 48
ReduceDataMinDist, 48
 Referenzaufruf, 61
 Referenzvektor, 11

- Regression, 9
 Regressionsanalyse, 9
 Regressionsfunktion
 erster Art, 2
 Regressionsproblem, 2
 Regressogramm, 112
 rekurrentes Netzwerk, 110
 relativierte Größe, 36
 resampling with replacement, → Ziehen mit Zurücklegen
 Residualgröße, 2
 Residuum, 2, 28, 29, 58, 91
 Reskalierung, 36
 rezeptives Feld, 11
 RMSE, → Wurzel der mittleren Fehlerquadratsumme
 Rohdaten, 35
 ROSENBLATT, 26
 Rückwärtsmethode, 36
 runoff, → Abfluss
-
- S**
- saisonale Komponente, 112
 Saisonbereinigung, 112
Salmo trutta fario, 68
 Sammon mapping, → Sammon-Abbildung
 Sammon-Abbildung, 15, 28, 57, 70, 78, 82, 84, 115
 scatter plot, → Streudiagramm
 Schicht
 Ausgabe-, 10
 Eingabe-, 10
 verdeckte, 10
 Schicht (eines KNNs), 10
 Schlitz (Stadt), 68, 81
 schrittweise Methode, 36
 Schrittweitensteuerung, 32
 Schüttung, → Abfluss
 selbstorganisierende Karte, 11
 self-organizing feature map, → selbstorganisierende Karte
 Sensitivität, 30
 Sensitivitätsanalyse, 39
 sensorische Karte, → selbstorganisierende Karte
 Sieger, 11
 Singulärwertzerlegung, 18, 33, 44, 52
 sinusoide Funktion, 32
 Skalarprodukt, 108
 Skalentransformation, 36
 Skalierung
 [0,1]-, 36, 69, 81
 ‘sliding window’-Ansatz, 68
 Softwarevoraussetzung, 59
 SOM, → selbstorganisierende Karte
 SOMAnalyze, 41, 48
 SOMCBVFileName, 48
 SOMHeight, 48
 SOMNumTrainingSteps, 48
 SOM_PAK (Software), 14, 16, 28, 51, 60, 62, 65, 66, 117, 128
 SOMReuse, 48
 SOMWidth, 48
 Spannweite, 28, 29, 89, 94
 Spline-Funktion, 17, 24, 43, 47
 Split-Sample-Validierung, 3
 SPSS (Software), 21, 94, 128
 Standardabweichung, 6
 Standardausgabe, 39, 51
 Standardisierung, 36
 Steinfliege, 69, 71, 121
 Stimuluswahl (SOM-Training), 12
 stochastische Abhängigkeit, 2, 77
 Störung, 78, 107
 Streudiagramm, 29
 string, 40, 62
 sum-of-squared errors, → Fehlerquadratsumme
 Superposition, 6, 18
 symbolischer Link, 38, 61
 Synapsenstärke, 9, 11
-
- T**
- t*-Verteilung, 21
 target, → Zielgröße
 Taylor-Entwicklung, 23
 TDS-Nr., → Testdatensatz-Nummer
 Testdaten, 3, 90
 für Abflussprognosen, 82
 Testdatensatz-Nummer, 82
 Theilkoeffizient, 105, 106, 112
 Thin-Plate-Spline, 17, 24, 43, 47
Thrips imiginis, 68
Tinodes rostocki, 70, 77
 Training, → Lernen
 Trainingsdaten, → Lerndaten
 Trajan (Software), 66, 82, 106, 128
 Trajektorie, 16, 28, 57, 85, 115
 Trichoptera, 69, 121
 TSP-Funktion, 17, 24, 43, 47

U

U-Matrix, 16, 28, 51, 57, 75, 83, 84, 115
 Überanpassung, 3, 25, 32
 Überlagerung, 6
 überwachtes Lernen, 10
 unabhängige Variable, 2
 univoltin, 79
 unsupervised learning, → unüberwachtes Lernen
 unüberwachtes Lernen, 10

V

Validierungsdaten, 3
 Validierungsdatensatz, 18
 Validity-Index-Netzwerk, 20
 Variable
 abhängige, 2
 erklärende, 2
 unabhängige, 2
 Variationsbreite, 111
 vector, 40
 Vektorquantisierung, 18
 VerboseLevel, 48
 verdeckte Schicht, 10
 Verfahren
 nicht-hierarchisches, 18
 partitionierendes, 18
 versteckte Schicht, → verdeckte Schicht
 Vertrauensintervall, → Konfidenzintervall
 Verzeichnisstruktur, 60
 VI-Netz, → Validity-Index-Netzwerk, 90
 Visu3DColX1, 48
 Visu3DColX2, 49
 Visu3DConstVector, 49
 Visu3DMaxX1, 49
 Visu3DMaxX2, 49
 Visu3DMeshLabelX1, 49
 Visu3DMeshLabelX2, 49
 Visu3DMeshLabelY, 49
 Visu3DMinX1, 49
 Visu3DMinX2, 49
 Visu3DNumGridPoints, 49
 visual (Software), 62, 70
 Visualisierungsmodus, 39
 Visualisierungsmöglichkeit, 114

Visualrbfn (Software), 33, 35–66, 68, 74, 75, 80, 82, 83, 95, 100, 106, 111, 112, 114, 116, 118–120, 123, 128
 visualrbfn.oct (Software), 38–41, 43, 50, 51, 59, 62, 146
 visusom (Software), 50, 60, 62, 64, 66, 97
 Vorhersageintervall, → Prognoseintervall
 Voronoi tessellation, → Voronoi-Parkettierung
 Voronoi-Diagramm, 12
 Voronoi-Parkettierung, 12
 Voronoi-Polygon, 112
 Vorverarbeitung
 von Daten, 35
 vorwärts gerichtetes Netzwerk, 10
 Vorwärtsmethode, 36, 82

W

Wadi, 110
 Wahrscheinlichkeitsdichte
 a-posteriori-, 22
 a-priori-, 22
 Wasserstand, 80
 Wavelet-basiertes KNN, 23
 Wetter-Abfluss-Modell, 80
 width, → Breite (einer Basisfunktion)
 Winnipeg-River, 110
 work_results.txt (Datei), 57, 58, 74, 90
 Wurzel der mittleren Fehlerquadratsumme, 3

Z

z-Transformation, 36
 Zeiger, 61
 Zeitfenster
 gleitendes, 68
 Zeitreihe
 multivariate, 81
 Zeitreihenanalyse, 9, 80
 Zelle
 eines Netzwerkes, 10
 Zentrum
 einer RBF, 17
 Zentrumsfunktion, → radiale Basisfunktion
 Ziehen mit Zurücklegen, 24
 Zielgröße, 2
 Zufallsvektor, 6
 Zuständigkeitsbereich, 11

Lebenslauf

Daten zur Person

Name: Obach
Vorname: Michael
Geburtsdatum: 15.08.1968
Geburtsort: Kassel
Familienstand: ledig
Staatsangehörigkeit: deutsch
Eltern: Karl Obach (Konstrukteur) und Anni Obach, geb. Salzman

Ausbildung

1974–1978 Grundschule Christian-Bitter-Schule, Melsungen
1978–1984 Gesamtschule Melsungen
1984–1987 Oberstufengymnasium Geschwister-Scholl-Schule, Melsungen
1987–1988 Grundwehrdienst in Gerolstein/Eifel und Schwarzenborn
1988 Beginn des Studiums der Mathematik (Diplomstudiengang) mit Nebenfach Biologie an der Universität Gesamthochschule Kassel
1991–1992 Berufspraktische Studien bei der Firma SEAT in Barcelona (Spanien)
1992 Vordiplom; Gesamtnote: *sehr gut*
1992 Beginn der Mitarbeit in der Forschungsgruppe Neuronale Netzwerke an der Universität Gesamthochschule Kassel
1998 Diplom in Mathematik; Abschlussnote: *sehr gut*
1998–2000 Beginn der Promotion und Arbeit an Forschungsprojekten an der Universität Kassel
2000–2003 Fortführung der Promotion und Forschungstätigkeit an der Limnologischen Fluss-Station Schlitz des Max-Planck-Institutes für Limnologie, Schlitz

Fremdsprachen, weitere Qualifikationen

- ▷ Englisch und Spanisch fließend,
Französisch Schulkenntnisse, Italienisch Grundkenntnisse
- ▷ Teilnahme an internationalen Kongressen
- ▷ wissenschaftliche Veröffentlichungen
- ▷ sehr gute Kenntnisse in angewandter Informatik

Adressen des Autors

Michael Obach

Universität Kassel:

Heinrich-Plett-Str. 40

(Raum 2409)

D-34132 Kassel

Tel.: +49-561-804-4376

Max-Planck-Institut:

Damenweg 1

D-36110 Schlitz

Tel.: +49-6642-9603-43

Privatadresse:

Spelzbachstraße 2

D-34212 Melsungen

E-Mail: mail@Michael-Obach.de
michaelo@neuro.informatik.uni-kassel.de
mobach@mpil-schlitz.mpg.de

WWW-URL: <http://www.Michael-Obach.de/>