Neuronale Netzwerkmodelle zur Analyse hochdimensionaler, multisensorischer Datensätze prozessierter Si-Wafer

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlichen Fakultät III –
Biologie und Vorklinische Medizin
der Universität Regensburg

vorgelegt von
Armin Schels
aus Lengfeld
2001

Promotionsgesuch eingereicht am: 12.09.2001

Die Arbeit wurde angeleitet von Prof. Dr. Elmar W. Lang

Prüfungsausschuss:

Vorsitzender: Prof. Dr. G. Hauska
1. Gutachter: Prof. Dr. E.W. Lang
2. Gutachter: Prof. Dr. U. Krey
Prüfer: Prof. Dr. E. Brunner

Inhalt

1. Einleitung

Ziel dieser Arbeit ist, verschiedene Netzwerkmodelle und –architekturen zur Analyse und Klassifikation hochdimensionaler, multisensorischer Datensätze zu untersuchen. Eine Aufgabenstellung besteht darin, Abnormitäten im hochdimensionalen Raum der multisensorischen nichtlinearen Zeitreihen zu detektieren und ggf. eine Klassifizierung zu ermöglichen. Die Datenanalyse muss aufgrund partiell fehlenden Vorwissens sowohl überwacht als auch mit unüberwachten Methoden durchgeführt werden. Um den „Fluch der Dimensionalität“ zu vermeiden, sowie systeminhärente Information zu detektieren, wird der hochdimensionale Datenraum reduziert bzw. signifikante Kennzahlen extrahiert, um anschließend eine einfache Klassifikation dieser Daten zu ermöglichen.
Ein weiteres Ziel der Arbeit besteht in der Analyse und Klassifizierung hochdimensionaler Datenräume ohne Dimensionsreduzierung. Konkret besteht die Aufgabe darin, mehrdimensionale Vektoren von Messergebnissen aufgrund eines Klassen-Labels zu gruppieren und neue Testdatensätze zu klassifizieren. Diese Klassifikationsergebnisse können für diagnostische Zwecke bei der Ursachenfindung von multivariaten Fehlerbildern verwendet und so ein KI-System aufgebaut werden, das unmittelbar nach der Detektierung nicht nur die Abnormität erkennt, sondern sogar die Ursache prognostizieren soll.

Das verwendete Datenmaterial stammt aus der Si-Halbleiterfertigung. Im Speziellen wurden Daten eines Logik-Bausteins verwendet, der bei Mobilfunk-Telefonen eingesetzt wird. Es handelt sich hierbei um eine Schaltung in Hybrid-Technologie, d.h. es sind sowohl digitale, als auch analoge Elemente in der Schaltung vorhanden. Je nach Produkt und Technologie werden einige hundert bis zu tausend Chips auf einem einzigen Wafer gefertigt (beim untersuchten Produkt ca. 500). Der Wafer ist eine monokristalline Silizium-Scheibe, auf der im Lauf der Fertigung, in einem Raster von ca. 1 cm Kantenlänge, die Chips prozessiert werden. Die Produktion ist sehr kosten- und zeitintensiv, so müssen die Wafer bis zu einigen hundert Prozessierungsschritte bis zum fertigen Produkt durchlaufen. Die Durchlaufzeit einer Scheibe von der Einschleusung bis zur Funktionsmessung beträgt, je nach Komplexität des Bauteils, mehrere Wochen. Die Waferausbeute liegt derzeit zwischen 43 und 93 Prozent und liefert damit noch ausreichend Verbesserungspotential.

Eine typische Abfolge von Prozessierungsschritten in der Halbleiterproduktion ist: Aufwachsen einer neuen halbleitenden Schicht, Strukturierung mittels Fototechnik, selektives Abtragen durch Nass- bzw. Trockenätzverfahren und die Dotierung des Materials durch Implantation. Nach der Fertigprozessierung der Wafer werden eigens aufgebrachte

Standardstrukturen, einfache Transistoren, Schichtwiderstände etc., gemessen und anschließend ein Funktionstest der einzelnen Bauteile durchgeführt. In dieser Arbeit werden sowohl die *in situ* aufgezeichneten Prozessdaten von Trockenätzanlagen, wie zeitlicher Temperaturverlauf, Signalintensitäten etc., als auch die nach der Fertigprozessierung erfassten Messwerte, untersucht. Ziel der Datenanalyse der einzelnen Prozessierungsschritte ist, Abhängigkeiten von einzelnen Observablen mit Ausbeute und Performance des Endproduktes zu erkennen, um so eine Fehlererkennung, Klassifikation und Fehlerdiagnose unmittelbar nach der Prozessierung einer einzelnen Scheibe zu ermöglichen.
Die statistische Prozesskontrolle der Halbleiterproduktion überwacht üblicherweise Messwerte, wie Schichtdicken oder Linienbreiten, die nach dem eigentlichen Einzelprozess stichprobenhaft gemessen werden. Der ursächliche Zusammenhang zwischen Prozessparametern und Prozessziel ist derzeit bei vielen Prozessen nicht ausreichend bekannt und verstanden. In der Realität kommt es deshalb oft vor, dass fehlprozessierte Scheiben erst bei der Funktionsmessung auffallen, was eine hohe Vergeudung von Fertigungskapazität bedeutet. Aus der Fehlererkennung und Diagnose der Prozessparameter erhofft man deshalb eine hohe Produktivitätssteigerung, sowie aufgrund der damit möglichen besseren Prozesskontrolle eine massive Qualitätsverbesserung.

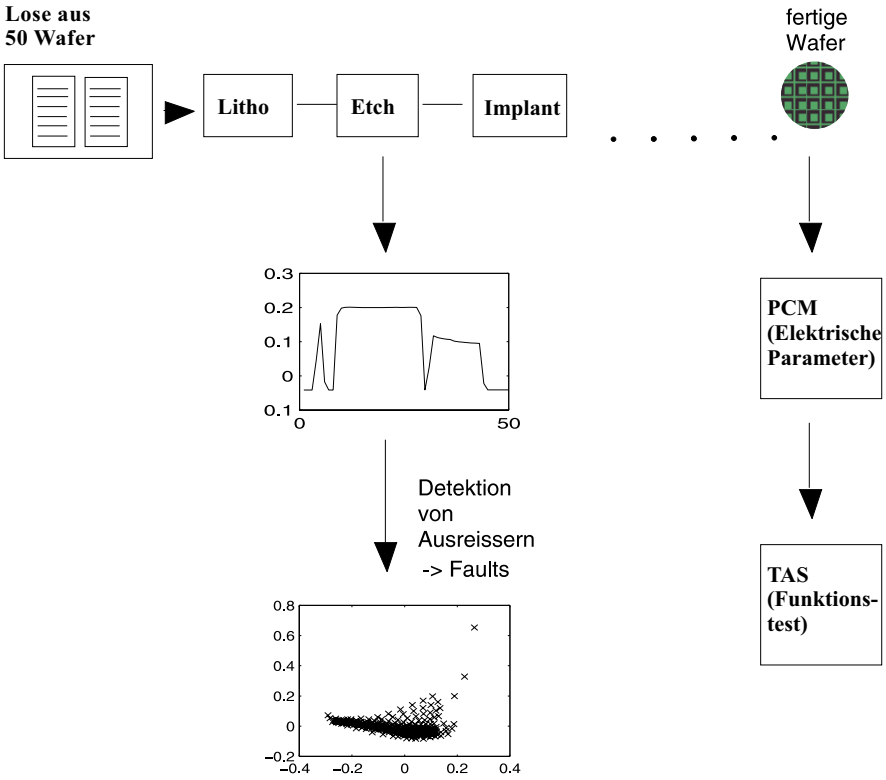

Abbildung 1.1: Grobe Skizze des Prozessierungsablaufes eines Loses in der Halbleiter-fertigung. Ein Los, bestehend aus 50 Wafer, durchläuft bis zu 400 Einzelprozesse. Während der Einzelprozesse werden einige physikalische Parameter als Funktion der Zeit aufgezeichnet. Am fertigen Wafer werden zunächst die PCM-Strukturen vermessen, anschließend wird der Funktionstest auf Bauteilebene durchgeführt.

Im zweiten Teil der Arbeit werden keine Zeitreihen von Prozessanlagen, sondern skalare Mess-Parameter untersucht. Diese im sogenannten Ritz-Rahmen aufgebrachten

Standardstrukturen wie Transistoren und Schichtwiderstände ermöglichen eine Beurteilung der Fertigung unabhängig vom eigentlichen Produkt und werden PCM-Parameter (*process control monitoring*) genannt. Da die PCM-Strukturen die gleichen Einzelprozesse wie die eigentlichen funktionalen Strukturen auf dem Wafer durchlaufen, sollten die Eigenschaften beider Strukturen korrelieren. Das heißt, falls die PCM-Strukturen eine gewisse Spezifikation erfüllen, sollte auch der Wafer eine entsprechende Ausbeute an funktionierenden Chips aufweisen.

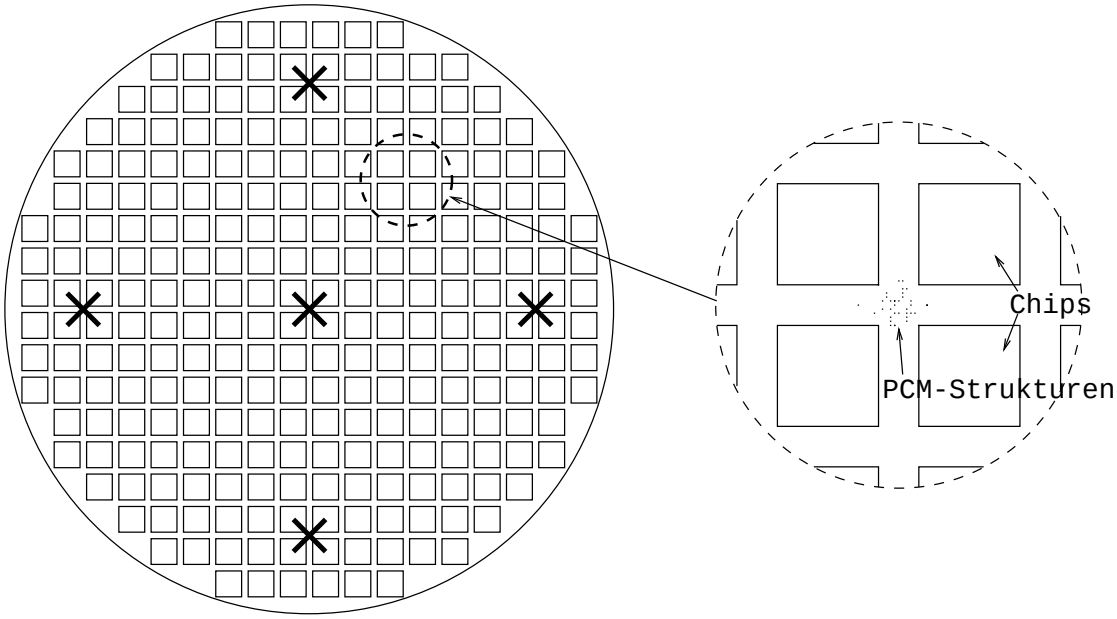

Abbildung 1.2: Schematisches Bild eines fertig prozessierten Wafers mit den aufgebrachten PCM-Strukturen. An den markierten Stellen werden die PCM-Parameter gemessen.

Da die eigentlichen Funktionsmessungen der Bauteile (ca. 1000 verschiedene Funktionstests) sehr aufwendig und teuer sind, werden an jedem Wafer an fünf Stellen (oben, unten, rechts, links, mitte) ca. 65 PCM-Strukturen gemessen. Es werden also pro Wafer zunächst 65 x 5 ≈ $3x10^2$ statt 500 x 1000=$5x10^5$ Messungen vorgenommen. Liegen die PCM-Werte innerhalb der Spezifikationsgrenzen, werden die Funktionsmessungen durchgeführt.

Diese Arbeit ist folgendermaßen aufgebaut: Nach der theoretischen Behandlung der Grundlagen neuronaler Netze und der verwendeten statistischen Verfahren in Kapitel 2 werden im dritten Kapitel die Analysen und Simulationsergebnisse dargestellt. Da im Zuge der Prozessierung die Zuordnung der Zeitreihen zu den Funktionsmessergebnissen verloren geht, wird in Kapitel 3.1 mit unüberwachten Methoden der Datenanalyse gearbeitet. Für ein bestimmtes Problem war die Information über Signalverlauf und zugehöriges Resultat bekannt. In Kapitel 3.2 werden diese Daten mit überwachten Analysemethoden untersucht und Informationen über die Zeitreihen mit dem jeweiligen Prozessziel erarbeitet. Kapitel 3.3 unterscheidet sich hinsichtlich des Datenmaterials von den vorherigen Kapiteln. Hier werden nicht multisensorische Zeitreihen, sondern hochdimensionale Testdaten mit *a-priori*-Information zur Mustererkennung und Klassifikation untersucht. Im letzten Kapitel werden die Ergebnisse kurz zusammengefasst.

2. Theorie und Grundlagen

Aufgrund der zunehmenden Rechenleistung wurde in den letzten Jahren eine Vielzahl neuer statistischer Verfahren zur Analyse multivariater Datensätze entwickelt und untersucht. Selbstadaptive Verfahren ermöglichen mit relativ hohem Wirkungsgrad eine passende Datenvorverarbeitung, Kennzahlenextraktion und Dimensionsreduzierung zur Datenvoranalyse. Unüberwachte Methoden werden zur Auswertung von Datensätzen ohne a-priori-Klasseneinteilung und damit zur Detektierung systeminhärenter Information zur Zustandsidentifikation eingesetzt. Ist Vorwissen über eine *a-priori*-Klassenteilung vorhanden, werden neuronale Netze mit überwachten Lernregeln zur Klassifizierung und Diagnose verwendet.
In diesem Kapitel werden nach den grundlegenden Konzepten künstlicher neuronaler Netze die speziellen Lernregeln und Analysemethoden vorgestellt, die in dieser Arbeit verwendet werden.

2.1 Grundlegende Konzepte

2.1.1 Entwicklung künstlicher neuronaler Netze

Die Arbeit an künstlichen neuronalen Netzen wurde motiviert durch neue Erkenntnisse über den Aufbau des menschlichen Gehirns. Das Gehirn ist ein komplexes in höchstem Maße parallel arbeitendes System. Bis zu Beginn des 20. Jahrhunderts, als längst bekannt war, dass sich der gesamte Körper in „atomistischer“ Struktur aus einzelnen Zellen aufbaut, glaubten die Wissenschaftler, das Gehirn bestehe eher aus einer Art amorpher Masse. Erst durch die fundamentalen Erkenntnisse von Golgi und Ramòn y Cajal wurde gezeigt, dass sich auch das Gehirn aus elementaren Funktionseinheiten, den sogenannten Neuronen, zusammensetzt.

Ein typisches Neuron des menschlichen Kortex besteht aus einem Zellkörper, der den Zellkern enthält, und einer Reihe wurmartiger Fortsätze, den Dendriten, und einem Axon. Die Dendriten dienen als Sensoren zum Sammeln von Informationen von anderen Neuronen. Das Axon wird im menschlichen Gehirn einige Zentimeter lang, ist am Ende verzweigt und endet

in den sogenannten Synapsen, die den Kontakt zu den Dendriten anderer Neuronen erlauben. Die Signalrichtung ist hauptsächlich von den Dendriten über den Zellkern zum Axon.

Heute ist der Aufbau, die chemischen Zusammenhänge und Funktionsweisen des Gehirns im mikroskopischen Bereich relativ gut erforscht. Die makroskopische Funktionsweise des Gehirns ist jedoch aufgrund der Komplexität des Systems bisher nur in Ansätzen erklärbar.
Aktuelle Schätzungen gehen davon aus, dass das Gehirn bereits im Frühstadium des menschlichen Embryos aus etwa 10^{14} Neuronen besteht, die durch Synapsen miteinander vernetzt sind [Thomson92]. Im Laufe der ersten Lebensjahre bilden sich etwa eine Million Synapsen pro Sekunde. Aus dieser starken Vernetzung und Parallelität resultiert die enorme Kapazität des Gehirns, die Fähigkeit zur Adaption und zur Lösung höchstkomplexer Aufgaben.

Die Entwicklung künstlicher neuronaler Netze in Anlehnung an das menschliche Gehirn begann Mitte des 20. Jahrhunderts. Als sich mit der Erfindung des Transistors ein enormer Fortschritt in der Rechenleistung von Elektronen-Gehirnen abzeichnete, glaubten Forscher euphorisch, dass künstliche neuronale Netze irgendwann in der Lage sein würden, jede ihnen gestellte Aufgabe zu erlernen und bravourös zu meistern. Heute werden die Fähigkeiten und Perspektiven neuronaler Netze weitaus realistischer gesehen.

2.1.2 Prinzipien des Lernens

Im Bereich der künstlichen neuronalen Netze unterscheidet man verschiedene Lernarten. Prinzipiell liegt die Lernfähigkeit eines neuronalen Netzwerkes in der langfristigen Dynamik der synaptischen Plastizität der Verbindungen zwischen den einzelnen Neuronen. Ein lernendes System versteht es dabei, aus vergangenen Erfahrungen, Schlüsse zu ziehen und sich mit diesen zu verändern.
Nach [Haykin99] ist Lernen wie folgt definiert:
Lernen ist ein Vorgang, bei dem die freien Parameter eines neuronalen Netzes nach bestimmten Regeln adaptiert werden, wobei laufend Eingabedaten (sogenannte Trainingsdaten) von außen an die Eingabeschicht des Netzes angelegt werden.

Es wird zwischen zwei grundlegend verschiedenen Arten von Lernen, dem unüberwachten bzw. überwachten Lernen, unterschieden.

Beim unüberwachten Lernen lernt ein System von einer Menge von Eingabevektoren ohne bestimmte Klassenzugehörigkeit. Ein bekanntes Beispiel dafür ist die Komprimierung von Bildern oder Tönen in der Telekommunikation für die Fernübertragung. Das System sucht sich hier wiederkehrende Muster oder Eigenschaften im Eingaberaum in selbst-organisierter Art und Weise. Es erarbeitet so eine Art Codebuch, das sowohl dem Sender als auch dem Empfänger bekannt sein muss. Neue Signale werden dann kodiert indem man einfach den Index des nächstgelegenen Codebuchvektors überträgt. Es gilt dabei einen Kompromiss zwischen minimalem Informationsverlust und möglichst starker Kompression zu finden.

Beim überwachten Lernen handelt es sich um die Klassifikation, also Zuordnung eines Satzes von Signalen und zugehörigen Klassen-Labels. Das heißt, zu jedem präsentierten Muster wird auch eine Soll-Ausgabe vorgegeben. Beim Training des Netzwerkes wird die Differenz zwischen der aktuellen Netzwerkausgabe und dem Soll-Ausgabewert bestimmt und modifiziert die synaptischen Verbindungen so, dass diese Differenz gegen Null konvergiert. Ein Spezialfall des überwachten Lernens stellt das „Reinforcement Learning“ dar. Hier kennt man zwar den korrekten Ausgabewert nicht, man weiß allerdings, ob das Ergebnis des neuronalen Netzes richtig oder falsch ist.

2.1.3 Perzeptrons

Das Perzeptron ist eines der einfachsten und ältesten Neuronen-Modelle. Es wird ausschließlich für überwachtes Lernen eingesetzt. Ein einzelnes Perzeptron kann genau eine Klasse beschreiben. Weiterhin kann das Perzeptron nur Entscheidungsgrenzen in Form einer Hyperfläche des $\mathbb{R}^n$ finden. Es ist auf solche Muster beschränkt, die linear separabel sind (siehe Abbildung 2.1). Sind mehrere Klassen vorhanden, müssen mehrere Perzeptrons parallel verwendet werden.

Im Folgenden wird zuerst ein einfaches, anschließend ein mehrschichtiges Perzeptron behandelt. Ein einfaches Perzeptron besteht aus einer Eingabeschicht, gewichteten Verbindungen w_{ij} zum Ausgabeneuron, einer Summations-Einheit Σ und einer Aktivierungsfunktion $g(\cdot)$. In manchen Arbeiten wird zudem eine bestimmte nichtlineare Ausgabefunktion vorgeschlagen. Üblicherweise wird dann entweder als Aktivierungsfunktion oder als Ausgabefunktion die identische Abbildung verwendet. In dieser Arbeit wird nur eine nichtlineare Funktion verwendet und die Begriffe Aktivierungsfunktion und Ausgabefunktion als Synonyme benützt.
Die Eingabedaten werden im Folgenden zu einem n-komponentigen Vektor $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, die m-dimensionalen Ausgabedaten zu $\boldsymbol{y} = (y_1, \ldots, y_m)^T$ und die Gewichtsvektoren zu $\boldsymbol{w}_j = (w_{j1}, \ldots, w_{jn})^T$ zusammengefasst.

Das Perzeptron bildet die gewichtete Summe des Eingabevektors gemäß:

$$y_j(\mathbf{x}) = g\left(\sum_{i=1}^{n} w_{ij} \cdot x_i + \theta_j\right) \qquad (2.1)$$

Wobei θ_j den Offset darstellt, der eine Entscheidungsebene ermöglicht, die nicht durch den Koordinatenursprung verläuft. Im Folgenden ist der Offset integriert, indem der Gewichtsvektor um ein Element $w_{j0} = \theta$ und die Eingabevektoren um ein festes Element $x_0 = +1$ erweitert wird.
Gleichung (2.1) lässt sich damit schreiben als:

$$y_j(\mathbf{x}) = g\left(\mathbf{w}_j^T \mathbf{x}\right) \qquad (2.2)$$

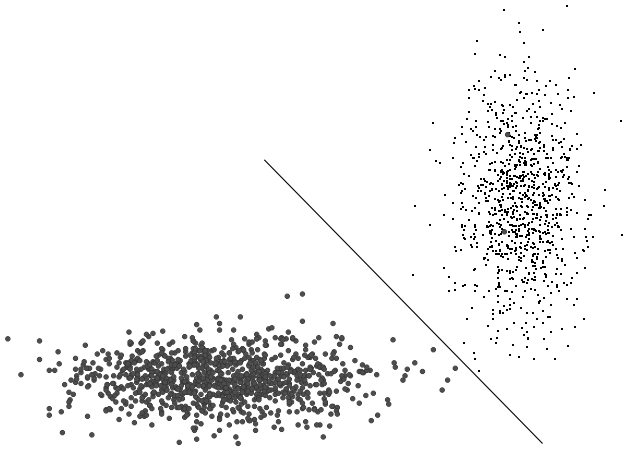

Abbildung 2.1: Ein Perzeptron lernt die Entscheidungsgrenze für zwei linear separable Klassen an Beispielen aus dem $\mathfrak{R}^2$. Nach 10000 Trainingszyklen ordnet es alle Trainingsmuster korrekt ein. Die Lernrate η war in diesem Beispiel konstant gleich 0,5. Rücken die Zentren der Gauß-Verteilungen noch näher zusammen, so dass Punkte einer Klasse im Gebiet der anderen liegen, muss η im Laufe des Trainings nach einem „Abkühlschema" erniedrigt werden um eine möglichst gute Entscheidungsregion zu finden.

Die Matrix der synaptischen Gewichte, die Gewichtsmatrix $\boldsymbol{W}$, stellt den freien Parametersatz eines neuronalen Netzes dar. Die Aktivierungsfunktion $g(\cdot)$ schränkt im wesentlichen die Bandbreite der Ausgabe ein. Gebräuchliche Aktivierungsfunktionen sind die Heaviside-Step-Funktion oder die Sigmoide-Funktion (siehe Abbildung 2.2). Gemeinsam ist allen Aktivierungsfunktionen, dass sie (meist monoton) steigend sind.
Gleichzeitig bringt die Einführung einer Aktivierungsfunktion ein nichtlineares Element in die Übertragungscharakteristik des Perzeptrons ein und erlaubt so im Falle mehrschichtiger Netze das Erlernen nichtlinearer Entscheidungsregionen zur Klassifizierung eines Datensatzes.

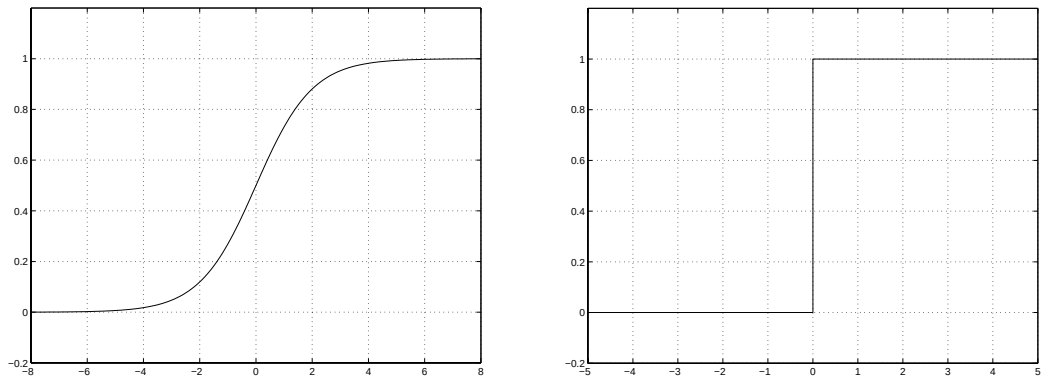

Abbildung 2.2: Gebräuchliche Aktivierungsfunktionen für lineare Perzeptrons. Gezeigt sind die Sigmoide- und die Heaviside-Step-Funktion.

Der Perzeptron-Lernalgorithmus zählt zu den überwachten Verfahren. Während des Lernens wird dem Perzeptron zu jedem Muster $\boldsymbol{x}$ die jeweilige Klassenzugehörigkeit als Soll präsentiert.

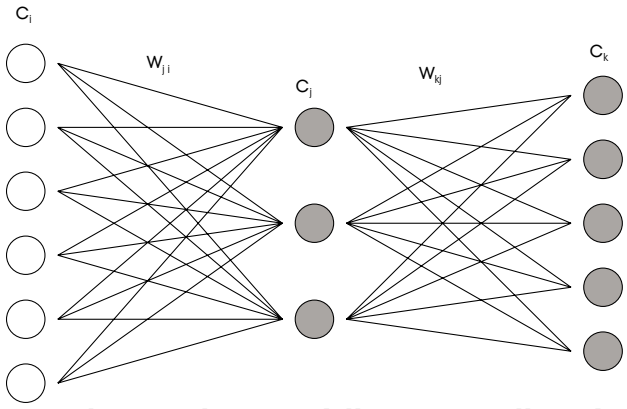

Abbildung 2.3: Schematisches Modell eines vollständig vernetzten mehrschichtigen Perzeptrons mit einer verdeckten Schicht. Die grau hinterlegten Neuronen sind einfache Perzeptrons während die weißen Einheiten im wesentlichen die „Dendriten" der verdeckten Schicht darstellen. Die Verbindungen sind mit Gewichten belegt. Die Komplexität der möglichen Entscheidungsregionen ist vor allem von der Anzahl an Neuronen in der verdeckten Schicht abhängig.

Der Perzeptron-Algorithmus sieht folgendermaßen aus:

1. Initialisiere den Gewichtsvektor ***w*** zufällig; wähle beispielsweise den Vektor eines beliebigen Musters aus. Wähle eine anfängliche Lernrate $\eta_0<1$. Setze den Zeitparameter $t=0$.
2. Ziehe zufällig ein Muster-Label-Paar $(\boldsymbol{x},d) \in S$ aus den Trainingsdaten.
3. Berechne die Aktivität des Neurons:
$$y\ (t) = g\left(\mathbf{w}^T \mathbf{x}\right)$$
4. Berechne den momentanen Fehler $e(t) = d - y(t)$
5. Belerne das Gewicht des Neurons gemäß der Delta-Regel:
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta(t) \cdot e(t) \cdot \mathbf{x}$$
6. Senke die Lernrate η, fahre fort mit Schritt 2 bis alle Muster richtig klassifiziert werden, eine vorher festgelegte Zahl an Lernschritten, oder ein anderes Abbruchkriterium, erreicht ist.

Für linear separable Klassen konvergiert der Algorithmus schon für $\eta(t)=const=1$ [Haykin99]. Im realitätsnäheren Fall nicht vollständig linear separabler Klassen muss die Lernrate im Laufe des Lernprozesses abgesenkt werden, so dass mit der Zeit die Muster immer geringeren Einfluss auf die Gewichte und somit die Entscheidungsregionen haben. Man hofft, dass damit das Perzeptron in einem möglichst optimalen Zustand einfriert.

Für komplexere Fälle, wie das bekannte XOR-Problem, versagt das einfache Perzeptron. Um nun komplexere, oder nichtlineare, Entscheidungsregionen zu ermöglichen, kann man zwischen Ein- und Ausgabeschicht noch eine oder mehrere verdeckte Schichten einbringen. Abbildung 2.3 zeigt das Modell eines mehrschichtigen Perzeptrons mit einer verdeckten Schicht. Diese Klasse neuronaler Netze ist prinzipiell in der Lage, beliebig komplexe Entscheidungsregionen zu finden, allerdings wird der rechnerische Aufwand sehr schnell erheblich. Als Trainingsverfahren wird meist Backpropagation eingesetzt (siehe Kapitel 2.3.1).

Die weiter unten besprochenen RBF-Netze (siehe Kapitel 2.2.4) können ebenfalls beliebig komplexe Entscheidungsregionen finden, allerdings unter wesentlich geringerem Rechenaufwand. Weiterhin ist die Komplexität der möglichen Entscheidungsregionen bei mehrschichtigen Perzeptrons im wesentlichen durch die Anzahl an Neuronen der verdeckten Schicht gegeben. Diese muss *a-priori* festgelegt werden.

2.1.4 Vektorbasierte neuronale Netze

Ein vektorbasiertes neuronales Netz besteht aus einer Menge A von Neuronen c_i, zusammen mit einer Menge $C = \{k_i\} \in A \times A$ von Kanten bzw. lateralen Verbindungen jeweils zweier Neuronen innerhalb einer Schicht. Diese Kanten sind nicht gewichtet und dienen ausschließlich dazu, eine topologische Struktur, also Nachbarschaftsbeziehungen der Neuronen, zu definieren. Prominenteste Vertreter solcher Netze sind die Kohonenkarte, die neuronalen Gase und die große Klasse der RBF-Netze (Radial Basis Function), die eine Art Hybrid-Ansatz aus Perzeptrons und vektorbasierten Netzen darstellen. Vektorbasierte neuronale Netze unterscheiden sich von Perzeptrons in folgenden Punkten:

Der Gewichtsvektor eines Neurons der verdeckten Schicht des RBF-Netzes hat eher die Bedeutung eines Referenzvektors $\boldsymbol{q}$ und wird im Folgenden so genannt.

Die Charakteristik eines Perzeptrons ist skalarprodukt-basiert, während vektorbasierte neuronale Netze auf die Ähnlichkeit ihres Gewichtsvektors zu einem Muster ansprechen. Diese ist im allgemeinen durch die euklidische Distanz zum Muster gegeben. Für die Aktivität eines RBF-Neurons c_i mit Referenzvektor $\boldsymbol{q}$ gilt:

$$\Phi_i(\mathbf{x}) = f\left(\|\mathbf{x}-\mathbf{q}\|\right) \tag{2.3}$$

wobei f eine zum Nullpunkt radialsymmetrische, im allgemeinen auf einen kleinen Bereich um den Nullpunkt beschränkte Funktion ist. Die am häufigsten verwendete Aktivierungsfunktion ist die Gauß'sche Glocke:

$$f(\mathbf{x}) = exp\left(-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{\sigma^2}\right) \tag{2.4}$$

Das rezeptive Feld eines RBF-Neurons ist lokal und i.d.R. radialsymmetrisch um seinen Referenzvektor. Im Gegensatz dazu ist das rezeptive Feld eines Perzeptrons global und unterteilt den $\mathfrak{R}^n$ in zwei Halbräume (siehe Abbildung 2.4).

Unter dem Gewinner $s(\boldsymbol{x})$ verstehen wir dasjenige Neuron einer Schicht von RBF-Neuronen dessen Referenzvektor $\boldsymbol{q}$ nach dem gegebenen Maß die höchste Ähnlichkeit zu einem Muster aufweist:

$$s(\mathbf{x}) = \underset{c_i \in A}{min}\|\mathbf{q}_i - \mathbf{x}\| \tag{2.5}$$

Dabei ist *A* die Menge aller Neuronen in der Schicht. Als Ähnlichkeitsmaß wird meist die euklidische Distanz verwendet.

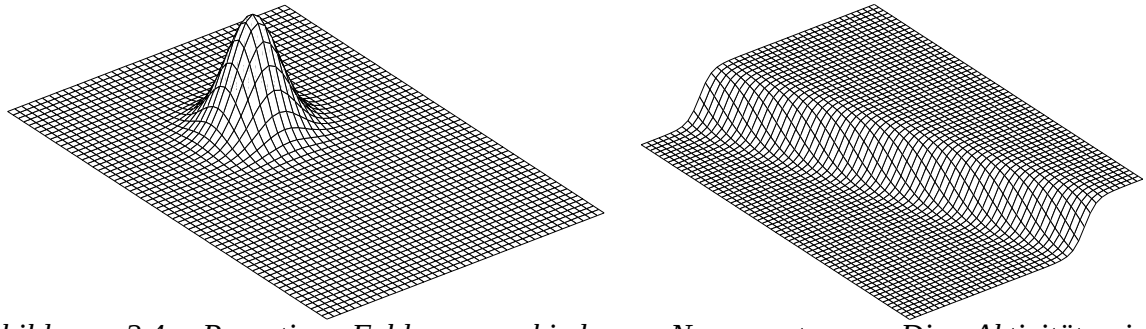

Abbildung 2.4: Rezeptive Felder verschiedener Neuronentypen. Die Aktivität eines vektorbasierten RBF-Neurons ist lokal während die eines Perzeptrons den Raum in zwei Halbräume teilt.

2.1.5 Voronoi-Tesselation und Delaunay-Triangulation

Der Begriff der Voronoi-Tesselation bzw. der Voronoi-Region ist sehr wichtig für das Verständnis vektorbasierter neuronaler Netzwerke. Man nehme eine Menge von Neuronen mit Referenzvektoren $Q = \{\boldsymbol{q}_i\} \subset \mathfrak{R}^n$. Dann ist die Voronoi-Region V_i des Vektors $\boldsymbol{q}_i$ definiert als die Menge aller Punkte $\boldsymbol{x} \in \mathfrak{R}^n$, deren euklidischer Abstand von $\boldsymbol{q}_i$ geringer ist, als von jedem anderen Vektor $\boldsymbol{q}_j \in Q$, für die also das Neuron c_i der Gewinner ist:

$$V_i = \left\{\mathbf{x} \in \Re^n \mid s(\mathbf{x}) = c_i\right\}$$

Man betrachtet nun die Voronoi-Regionen aller Einheiten *c*. Durch die entstehende Konstruktion der Voronoi-Tesselation wird der Raum in eine Anordnung von konvexen einfach zusammenhängenden Regionen aufgeteilt. Diese erinnern sehr stark an die Brioullin'schen Zonen der Festkörperphysik. Es gilt also

$$\bigcup_{c_i \in A} V_i = \Re^n$$

und

$$\left((\mathbf{x}_1 \in V_i) \wedge (\mathbf{x}_2 \in V_i)\right) \Rightarrow \left(\mathbf{x}_1 + \lambda(\mathbf{x}_2 - \mathbf{x}_1)\right) \in V_i \qquad \text{mit } 0 < \lambda \leq 1$$

Nachfolgend wird der Einfachheit halber als Voronoigebiet eines Neurons *c* das Voronoigebiet seines Referenzvektors $\boldsymbol{q}_c$ bezeichnet. Weiterhin bezeichnen wir mit

$$F_{c_i} = \left\{\mathbf{x} \in S \mid s(\mathbf{x}) = c_i\right\}$$

die Voronoimenge des Neurons c_i, d.h. die Menge aller Muster des Eingaberaumes, die in sein Voronoigebiet fallen. Verbindet man alle Punkte $\boldsymbol{q}_i \in Q$, deren Voronoigebiete eine

gemeinsame Kante aufweisen, so erhält man die sogenannte Delaunay-Triangulation. Abbildung 2.5 veranschaulicht beide Konzepte für den Fall eines zweidimensionalen Raums.

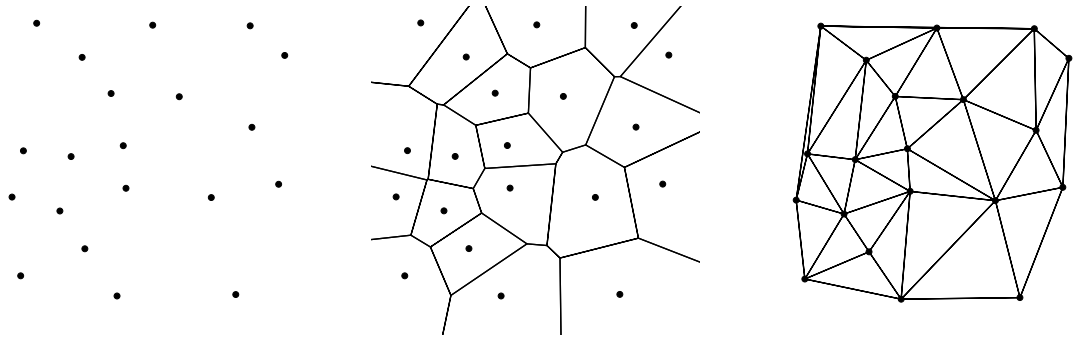

Abbildung 2.5: Darstellung der Voronoi-Tesselation anhand einer Menge an Punkten des $\Re^n$. Der Raum wird aufgeteilt in konvexe Regionen die jeweils aus der Punktmenge bestehen, die dem Referenzvektor am nächsten liegt (siehe [Fritzke98]). Die Delaunay-Triangulation verbindet solche Punkte, deren Voronoi-Region eine gemeinsame Kante hat.

2.2 Spezielle neuronale Netze

Dieses Kapitel stellt die theoretischen Grundlagen der speziellen neuronalen Netze vor, die in dieser Arbeit verwendet werden. Nach der Beschreibung der Kohonenkarten und der wachsenden neuronalen Gase werden die RBF-Netzwerke behandelt.

2.2.1 Die Kohonenkarte (SOM)

Die Kohonenkarte oder Self Organizing Feature Map (SOM) stellt ein vergleichsweise altes Verfahren dar [Kohonen82]. Sie fällt in die Klasse der unüberwachten Lernverfahren. In selbstorganisierter Art und Weise werden aus einem Satz $\{\boldsymbol{x}_i\} \in S$ von Daten im Eingaberaum gewisse Features extrahiert. Ein solches Merkmal wird repräsentiert durch einen bestimmten Vektor im Eingaberaum. Anschaulich gesprochen verteilt die SOM im Laufe des Trainingsprozesses eine festgelegte Zahl an Referenzvektoren auf solche Bereiche des Eingaberaumes, wo die Trainingsmuster dicht liegen. Der entscheidende Vorteil der SOM ist die Erhaltung der grundlegenden Topologie des Eingaberaumes, d.h. auf der fertig belernten Karte werden benachbarte Neuronen auch ähnliche Referenzvektoren haben [Zell97] (siehe Abbildung 2.6).

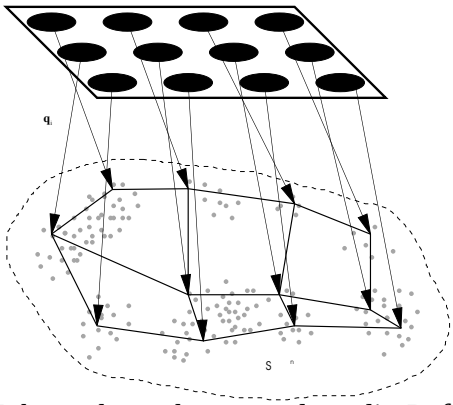

Abbildung 2.6: Die Kohonenkarte lernt so, dass die Referenzvektoren **q** *in Gebiete endlicher Wahrscheinlichkeitsdichte zeigen. Benachbarte Neuronen, also solche mit geringem euklidischen Abstand, haben damit ähnliche Referenzvektoren. Die netzartige Struktur im Eingaberaum entsteht, indem die Referenzvektoren benachbarter Neuronen durch Linien verbunden werden.*

In höherdimensionalen Räumen kann man mit ihrer Hilfe wichtige Merkmale der Eingabedaten finden und so ein Codebuch erstellen. Ihre Bedeutung liegt auch in der Möglichkeit der Visualisierung über die bloße Projektion auf zwei- bzw. dreidimensionale Schnitte hinaus.

Im Prinzip besteht die Kohonenkarte aus einer regelmäßigen Anordnung von Neuronen mit lokalen rezeptiven Feldern. Auf dem normalerweise zwei- oder dreidimensionalen Gitter

werden durch den unüberwachten Lernprozess Referenzvektoren im Eingaberaum platziert. Die Charakteristik eines Neurons c_j mit Referenzvektor $\boldsymbol{q}_j$ ist

$$\Phi_j(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{q}_j \right\| \tag{2.6}$$

Die Aktivität ist also nur abhängig vom euklidischen Abstand des Referenzvektors zum Eingabemuster. Damit fällt die Kohonenkarte in die Klasse der vektorbasierten Netze.

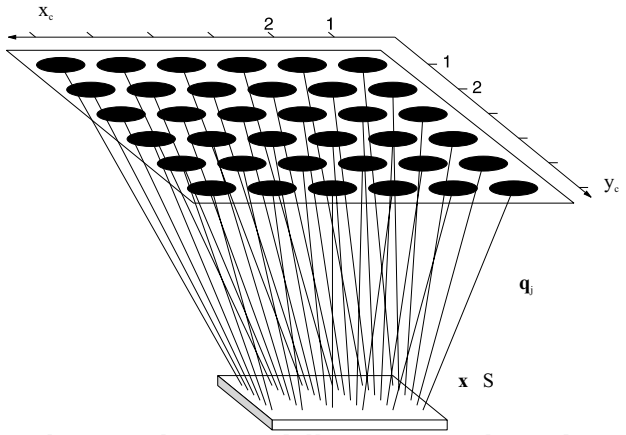

Abbildung 2.7: Schematisches Modell einer Kohonenkarte für den quadratischen zweidimensionalen Fall. Die Koordinaten $(x_c,y_c)_j$ *geben die Position des Neurons auf der Karte an.*

Um die genannte topologieerhaltende Abbildung zu realisieren, erfolgt der Lernprozess auf der Basis sogenannten weichen Wettbewerbslernens: Für jedes Muster $\boldsymbol{x}$ wird der Referenzvektor des Gewinners s, sowie die Neuronen in topologischer Nachbarschaft von s, in Richtung des Eingabemusters verschoben. Beim harten Wettbewerbslernen wird dagegen nur der Gewinner selbst belernt.
Der SOM-Algorithmus besteht im wesentlichen aus folgenden Schritten:

1. Lege die Dimensionalität (im allgemeinen Zwei) und Größe der Kohonenkarte fest.
2. Initialisiere die Elemente der Gewichtsvektoren $\boldsymbol{q}_i$ aller Neuronen zufällig innerhalb eines Intervalls $[rand_{min}, rand_{max}[$.
3. Beginne mit einer Lernrate $\eta=\eta_0$ und einer Breite $\sigma=\sigma_0$ der Nachbarschaftsfunktion und setze t=0.
4. Wähle zufällig ein Muster $\boldsymbol{x} \in S$.
5. Finde den Gewinner $s(\boldsymbol{x})$, also das Neuron c_i , in dessen Voronoigebiet das Muster $\boldsymbol{x}$ fällt.
6. Belerne die Gewichte aller Neuronen c_j gemäß

$$\mathbf{q}_j(t+1) = \mathbf{q}_j(t) + \eta(t) h_j(t)\left(\mathbf{x} - \mathbf{q}_j(t)\right) \qquad \forall j = 1,\ldots,N$$

7. Vermindere η und σ gemäß

$$\eta(t+1) = \eta(t) - \Delta_\eta$$
$$\sigma(t+1) = \sigma(t) - \Delta_\sigma$$

8. Fahre fort mit 4. bis eine vorher festgelegte Zahl an Lernschritten erreicht ist.

Als Nachbarschaftsfunktion wählt man im allgemeinen die Gauß'sche Glocke

$$h_j(t) = exp\left(-\frac{d_{js}^2}{\sigma^2}\right) \tag{2.7}$$

mit der euklidischen Distanz oder der, auch als Manhattan-Distanz bekannten, Norm

$$d_{js} = \left\|x_{cj} - x_s\right\| + \left\|y_{cj} - y_s\right\| \tag{2.8}$$

wobei die x_{cj}, y_{cj} den Positionen der Neuronen auf der Karte entsprechen. Der Effekt ist, dass das Siegerneuron am stärksten auf den Eingabevektor zu bewegt wird, seine direkten Nachbarn etwas schwächer. Alternativ kann der Algorithmus auch mit anderen Nachbarschaftsfunktionen verwendet werden. Prominentestes Beispiel ist hier die sogenannte Chief-Hat-Funktion, die konstant bis zu einem gewissen Abstand vom Siegerneuron gleich 1 ist und dann auf 0 abfällt. In den Simulationen konnte kein entscheidender Unterschied zwischen beiden Nachbarschaftsfunktionen festgestellt werden. Daher wurde durchgehend die Gauß'sche Nachbarschaftsfunktion verwendet.
Wichtig ist nur, dass zu Beginn ein großes Gebiet belernt wird, das im Laufe der Zeit kleiner wird. So kann sich am Anfang des Trainings die globale Ordnung der Karte ausbilden (Ordnungsphase), während später bei kleineren Nachbarschaften und Lernparametern die lokale Ordnung gebildet wird (Konvergenzphase).

Der Kohonen-Algorithmus ist sehr robust. Die Wahl der Parameter für die Lernrate η und die Breite σ der Nachbarschaftsfunktion h ist nicht sehr kritisch. Für η und σ werden im allgemeinen monoton fallende Funktionen gewählt. Von diesen hängt in erster Linie ab, wie gut die fertige Karte die Daten repräsentiert und vor allem, wie gut die Prämisse der Topologieerhaltung erfüllt wird. Fällt beispielsweise die Lernrate zu schnell im Vergleich zur Nachbarschaftsfunktion, kann die SOM in ein lokales Minimum fallen und in einem suboptimalem Zustand einfrieren. In theoretischen Untersuchungen wird dabei der momentane Quantisierungsfehler, oder die Shannon-Entropie, als Zustandsgröße definiert.

Entscheidend in allen Simulationen sind folgende Punkte:
Die Lernrate η sollte anfangs, während der sogenannten Ordnungsphase, wo sich die Topologie der Karte ausbildet, von der Größenordnung 1 sein. Später während der Konvergenzphase, sollte sie bis auf einen endgültigen Wert von etwa 0,01 sinken.

Die Breite σ muss anfangs sehr groß gewählt werden, so dass sich die Karte entfalten kann und sich die Nachbarschaftsbeziehungen vollständig ausprägen. Konkret sollte $\sigma(0)$ so gewählt werden, dass $h \approx 1$ für alle Neuronen der Karte beträgt.

Noch wichtiger ist, dass die Lernrate η im Verhältnis zu σ nicht zu schnell absinkt, sonst kann sich die globale Ordnung nicht ausbilden.

Die konkrete Wahl des Abkühlschemas hatte in allen Simulationen keinen großen Einfluss auf die Ausbildung einer topologisch korrekten Karte. Bei allen Simulationen wurde ein lineares bzw. quadratisches Abkühlschema verwendet.

2.2.2 Wachsendes neuronales Gas (GNG)

Die offensichtlichsten Nachteile der Kohonenkarte sind die starre Netzwerkgröße und die feste Dimensionalität des Netzes, die *a-priori* festgelegt werden müssen. Wünschenswert wäre ein System, das sich *online*, zur Laufzeit, kontinuierlich besser einer gegebenen Datenverteilung anpasst. Diese Anforderungen erfüllt das wachsende neuronale Gas (*Growing Neural Gas*) das von Fritzke auf der Basis des *neural gas algorithm* von Martinetz und Schulten [Fritzke95, Martinetz91] entwickelt wurde.

Das GNG stellt eine Methode dar, die kontinuierlich neue Einheiten einfügt. Dies geschieht indem jedem Neuron $c \in A$ eine Ressourcevariable E_c zugeordnet wird, die ein lokales Fehlermaß akkumuliert. Im einfachsten Fall wird für jedes präsentierte Muster $\boldsymbol{x}$ der Quantisierungsfehler, also der quadratische Abstand von Eingabevektor und Gewinner, zur Ressourcevariablen des Gewinners addiert:

$$\Delta E_s = \left\| \mathbf{x} - \mathbf{q}_s \right\| \tag{2.9}$$

Dann wird in definierten Abständen ein neues Neuron in der Nachbarschaft der Einheit mit maximaler Ressource eingefügt. Dadurch entstehen neue Einheiten nur in Regionen des Eingaberaumes, wo die Wahrscheinlichkeitsdichte endlich ist.
Das Lernverfahren beruht, wie bei der Kohonenkarte, auf weichem Wettbewerbslernen. Konkret wird für jedes präsentierte Signal der Gewinner und seine nächsten Nachbarn in Richtung des Musters verschoben, allerdings wesentlich schwächer als bei SOM. Schließlich werden bei SOM neue Regionen des Eingaberaumes nur „erobert“, indem bereits existierende Neuronen dorthin verschoben werden, während bei GNG ständig neue Neuronen in interessierenden Raumregionen eingefügt werden. Bereits existierende Neuronen ändern ihre Referenzvektoren nur minimal.

Die topologische Struktur des Netzwerkes ist definiert durch eine Menge an Kanten $C \subset A \times A$, die dynamisch generiert werden. Für jedes präsentierte Muster werden die beiden nächstliegenden Neuronen s_1, s_2 gesucht und mit einer Kante (s_1, s_2) verbunden. Dies garantiert die Erhaltung der Topologie des Eingaberaumes. Zwei Neuronen werden nur dann durch eine Kante verbunden, wenn sie ähnliche Referenzvektoren haben. Mit diesen Kanten ist eine Altersvariable *a* verknüpft. Solche Kanten, die ein gewisses Alter überschreiten, werden automatisch entfernt, ebenso wie Neuronen, die keine Kanten haben.
Eine neue Einheit wird nach einer festgelegten Anzahl von Schritten eingefügt. Dazu wird das Neuron *r* mit dem größten akkumulierten Fehler gesucht und das neue Neuron *f* in der Mitte zwischen *r* und dem Neuron *l* eingefügt, das unter seinen Nachbarn die größte Ressource hat. Die Ressourcevariablen von *r*, *l* und *f* werden so verteilt, dass die Summe der lokalen

Ressourcen gleich bleibt. Weiterhin werden zwei neue Kanten (*r*,*f*) und (*f*,*l*) geschaffen und die alte (*r*,*l*) entfernt.

Das Verfahren wird abgebrochen, wenn eine maximale Anzahl an Neuronen oder ein alternatives Kriterium erreicht ist. Konkret sieht der Algorithmus wie folgt aus:

1. Setze den Zeitparameter $t = 0$. Initialisiere die Menge A mit zwei Neuronen c_1 und c_2 deren Referenzvektoren zufällig aus den Mustern ausgewählt wird. Initialisiere die Menge *C* als leere Menge:
$$A = \{c_1, c_2\}, \quad C = \varnothing$$
2. Wähle zufällig einen Eingabevektor $\mathbf{x}$.
3. Bestimme Gewinner s_1 und zweitbeste Einheit s_2.
4. Verbinde s_1 und s_2 durch eine Kante, wenn sie nicht bereits existiert. Setze das Alter dieser Kante auf 0:
$$C = C \cup \{(s_1, s_2)\}$$
$$a_{(s_1,s_2)} = 0$$
5. Addiere den Quantisierungsfehler zur Fehlervariable von s_1:
$$\Delta E_{s_1} = \left\| \mathbf{x} - \mathbf{q}_{s_1} \right\|$$
6. Belerne die Referenzvektoren des Gewinners s_1 und seiner Nachbarn:
$$\Delta \mathbf{q}_{s_1} = \eta\left(\mathbf{x} - \mathbf{q}_{s_1}\right)$$
$$\Delta \mathbf{q}_j = \eta\left(\mathbf{x} - \mathbf{q}_j\right) \qquad \forall c_j \in N_{s_1}$$
7. Inkrementiere das Alter aller von s_1 ausgehenden Kanten:
$$a_{k_i} = a_{k_i} + 1$$
8. Entferne alle Kanten mit einem Alter größer a_{max}. Entstehen dabei Neuronen ohne Kanten entferne sie.
9. Ist *t* ein ganzzahliges Vielfaches einer festen Anzahl an Schritten ν, füge eine neue Einheit ein:
   - Finde das Neuron *r* mit maximaler Ressourcevariable und unter seinen Nachbarn aus N_r das Neuron *l* mit größter Ressourcevariable:
$$r = \max_{c \in A} E_c$$
$$l = \max_{c \in N_r} E_c$$
   - Erzeuge ein neues Neuron *f*. Initialisiere seinen Referenzvektor als arithmetisches Mittel von *r* und *l:*
$$\mathbf{q}_f = \frac{1}{2}\left(\mathbf{q}_r + \mathbf{q}_l\right)$$
$$A = A \cup f$$
   - Verringere die Ressourcevariablen von *r* und *l* um einen Faktor α und setze die Ressourcevariable von *f* als arithmetisches Mittel von *r* und *l*:
$$\Delta E_{r,l} = -\alpha E_{r,l}$$
$$E_f = \frac{1}{2}\left(E_r + E_l\right)$$

- Ersetze die alte Kante (*r*,*l*) durch zwei neue (*r*,*f*) und (*f*,*l*)

$$C = C \cup \{(r, f);(f, l)\} \neg (r, l)$$

10. Vermindere die Ressourcevariablen aller Neuronen um einen Faktor β. Durch diesen Schritt werden erst kürzlich akquirierte Ressourcen höher bewertet.

$$\Delta E_c = -(1-\beta)E_c \qquad \forall c \in A$$

11. Fahre fort mit Schritt 2 bis eine vorgegebene Anzahl an Neuronen erreicht ist, oder der während eines Zyklus von ν Schritten akkumulierte Quantisierungsfehler eine vorgegebene Grenze unterschreitet.

Die Vorteile des GNG sind, dass die Anzahl an Einheiten nicht *a-priori* festgelegt werden muss. Darüber hinaus ist es prinzipiell möglich, mit diesem System Cluster zu finden. Da die Kantenstruktur als Subgraph der Delaunay-Triangulation nur Neuronen über Gebiete endlicher Wahrscheinlichkeitsdichte hinweg verbindet, wird auch das neuronale Gas in Cluster zerfallen.
Die Struktur des Netzwerkes ist nicht, wie etwa bei Kohonenkarten, oder den ebenfalls von Fritzke [Fritzke93] eingeführten wachsenden Zellstrukturen, von vornherein auf eine bestimmte Dimensionalität begrenzt. Die Wahl der Parameter ist konstant und muss nicht während der Trainingsphase angepasst werden.
Das GNG kann auch nichtstationären Verteilungen folgen [Fritzke99]. Hierfür ist die Konstanz der Parameter entscheidend, da das System nicht „einfriert“, wie beispielsweise die SOM.

2.2.3 Überwacht wachsendes neuronales Gas (SGNG)

Das wachsende neuronale Gas erzeugt eine Art unbewerteten Codebuchs und findet Repräsentanten im Eingaberaum so, dass die Wahrscheinlichkeitsdichte der Muster im Eingaberaum gut approximiert wird. Will man allerdings mit diesem System Ressourcen zur Klassifizierung allozieren, beispielsweise die Zentren für ein RBF-Netzwerk oder die Stützpunkte für die in Abschnitt 3.3.5 diskutierten lokalen konstanten Abbildungen, ist dieser Ansatz nicht sinnvoll. Ein Ansatz, diese Schwächen auszugleichen und die Klassenlabel mit in den Trainingsprozess einzubeziehen, wurde von Fritzke für Klassifizierungsprobleme diskutiert [Fritzke94].

Er schlägt einen zweistufigen Trainingsprozess vor, bei dem gleichzeitig ein wachsendes neuronales Gas entwickelt und auf den Zentren ein RBF-Netz trainiert wird. Statt des Quantisierungsfehlers wird der lokale Klassifizierungsfehler als Ressourcevariable beim Gewinnerneuron akkumuliert. Damit werden neue Einheiten an solchen Stellen eingefügt, wo die Klassifizierung bisher schlecht ist.

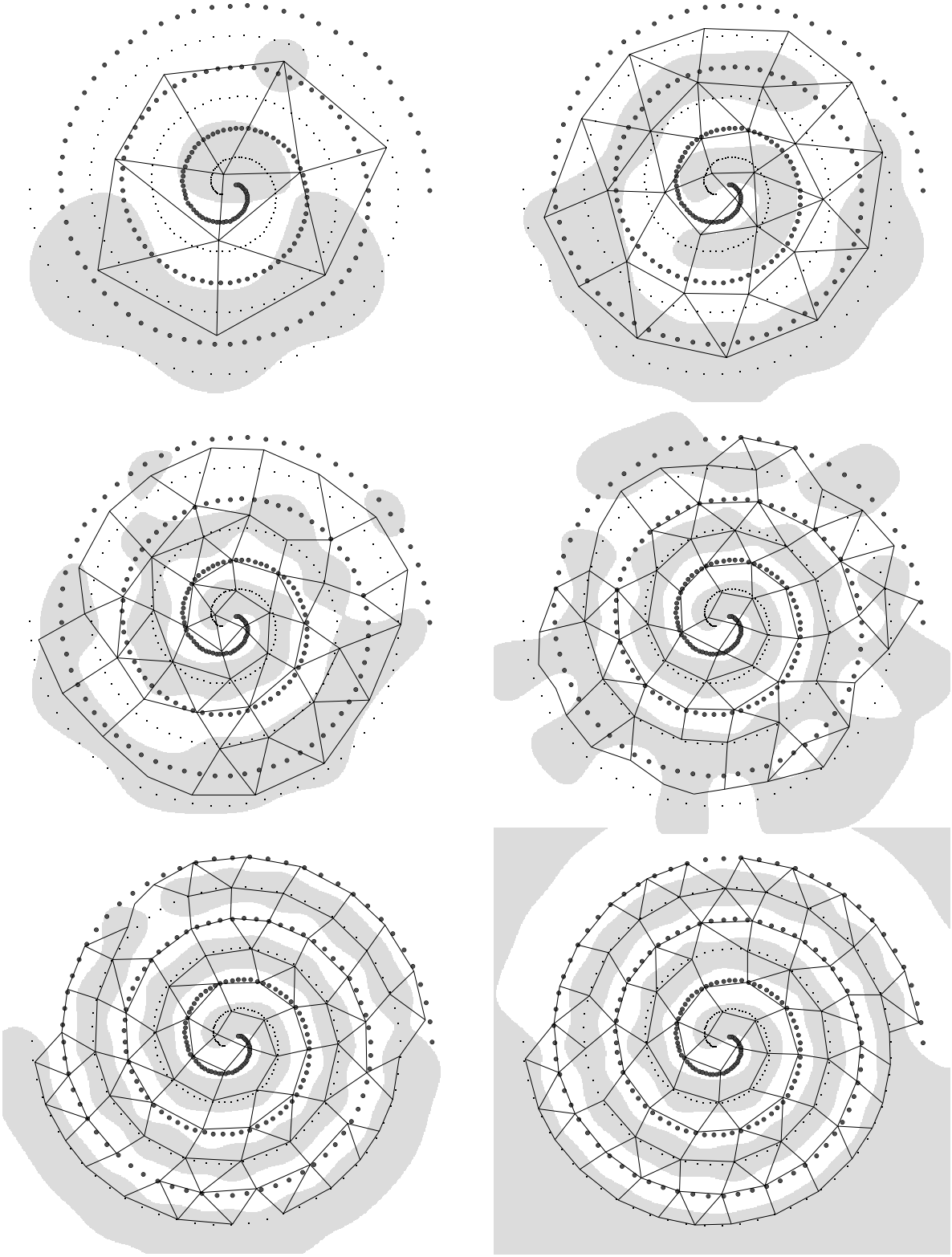

Abbildung 2.8: Entscheidungsregionen für das berühmte two-spirals-problem nach dem Einfügen von 10, 30, 50, 70, 85 und 95 Einheiten. Als Abbruchkriterium für den SGNG-Algorithmus wurde hier definiert, wenn beim Einfügen eines neuen Neurons die maximale Ressourcevariable einen Wert von 0,1 unterschreitet. Dies war bei 95 Neuronen der Fall. Korrekt klassifiziert wurden 209, 250, 290, 349, 393 bzw. 396 von insgesamt 400 Datenpunkten.

Dieser Ansatz wurde von Leim [Leim00] dahingehend erweitert, dass er die Diskretisierung des Klassenlabels durch eine kontinuierliche Darstellung ersetzte. Dieser Schritt stellt eine interessante Erweiterung dar, da speziell bei den zu untersuchenden Daten in Kapitel 3, keine diskrete Klasseneinteilung *a-priori* feststeht und dadurch eine willkürliche und evtl. falsche Klasseneinteilung erfolgt. Dazu wurde der Algorithmus wachsender neuronaler Gase folgendermaßen erweitert:

1. Jedem Neuron c_i des GNG wird zusätzlich noch ein Label θ zugeordnet, der den mittleren Klassenlabel in der momentanen Voronoi-Region V_{ci} des Neurons annähern soll. Bei der Initialisierung werden die ersten beiden Einheiten schlicht mit dem Label der entsprechenden, zufällig gezogenen, Muster versehen.

2. Es wird eine weitere Lernrate η_θ eingeführt. In Simulationen zeigte sich, dass diese gleich der Lernrate η_s gesetzt werden kann. Bei jedem Zyklus wird zusätzlich der Label θ_{s1} des Gewinnerneurons in Richtung des Datenlabels belernt. d stellt wieder den Label des gerade präsentierten Musters dar:
$$\Delta\theta_{s_1} = \eta \cdot \left(d - \theta_{s_1}\right)$$

3. Schritt 5 des GNG-Algorithmus wird ersetzt durch
$$\Delta E_{s_1} = \left\| d - \theta_{s_1} \right\|$$

Bei jedem Lernschritt wird also beim Gewinnerneuron eine Größe addiert, die ein Maß für die lokale Varianz des Klassenlabels darstellt. Damit werden neue Neuronen in solchen Gebieten eingefügt, wo die Schwankungen im Klassenlabel der Eingabedaten hoch sind.

Die diskutierten Verfahren der wachsenden neuronalen Gase eignen sich besonders gut um Ressourcen, d.h. insbesondere die Zentren bzw. Stützstellen für RBF-Netze, zu allozieren. Die RBF-Netze werden im folgenden Abschnitt besprochen.

2.2.4 RBF-Netze

RBF-Netze (Radial Basis Function) fallen in den Bereich überwachter Lernverfahren und eignen sich besonders gut für hochkomplexe Klassifizierungsaufgaben. Sie sind wie mehrschichtige Perzeptrons in der Lage, nichtlineare Entscheidungsgrenzen zu finden. Die RBF-Netze haben jedoch den entscheidenden Vorteil, dass sie wesentlich schneller lernen, in ihren Anwendungen flexibler sind und im allgemeinen besser generalisieren [Haykin99]. Prinzipiell lässt sich mit RBF-Netzen jede Abbildung beliebig genau nähern.

Die Struktur eines RBF-Netzes zeigt Abbildung 2.9. Es besteht im wesentlichen aus zwei Schichten. Die erste Schicht mit Neuronen c_j ist mit lokalen rezeptiven Feldern aufgebaut, die wie die Kohonenkarte auf gewisse Merkmale im Eingaberaum ansprechen. Die Ausgabeneuronen dieser ersten Schicht sind mit einer zweiten Schicht von Neuronen c_k vollständig vernetzt, die nichts anderes als einfache Perzeptrons darstellen. Die Anzahl dieser Perzeptrons entspricht der Anzahl an Klassen des diskreten Klassifizierungsproblems.

Die Aktivierungsfunktion der Neuronen in der verdeckten Schicht ist radialsymmetrisch um seinen Referenzvektor $\boldsymbol{q}$. Im allgemeinen wählt man die Gaußfunktion so, dass für die Aktivität dieser Neuronen c gilt:

$$\Phi_j(\mathbf{x}) = exp\left(-\frac{\|\mathbf{q}_j - \mathbf{x}\|^2}{\sigma_j^2}\right) \tag{2.10}$$

Verwendet man als Ausgabefunktion die identische Abbildung, ergibt sich die Ausgabeaktivität der Neuronen c_k zu:

$$y_k(x) = \sum_{j=1}^{R} w_{kj}\Phi_j(\mathbf{x}) + \theta_j = \sum_{j=1}^{R} w_{kj}\, exp\left(-\frac{\|\mathbf{q}_j - \mathbf{x}\|^2}{\sigma_j^2}\right) + \theta_j = \mathbf{w}_k^T \boldsymbol{\Phi}(\mathbf{x}) \tag{2.11}$$

mit R: Anzahl der Neuronen in der verdeckten Schicht
$\theta = w_{k0}$
$\Phi_0(\boldsymbol{x}) = 1$

Die Aktivität der Ausgabeneuronen soll 1 sein, wenn das Muster zur Klasse k gehört, sonst 0:

$$d_k(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in Y_1 \\ 0 & \text{sonst} \end{cases} \tag{2.12}$$

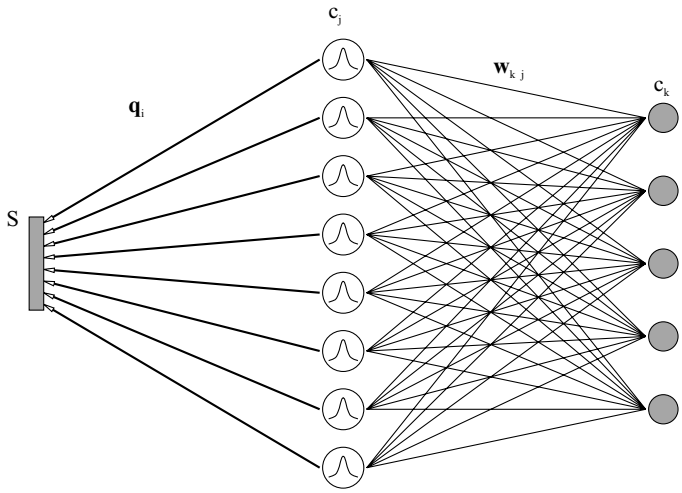

Abbildung 2.9: Schematische Darstellung eines RBF-Netzwerkes. Die Gauß-Kurve in den Neuronen der ersten Schicht soll veranschaulichen, dass es sich hier um Neuronen mit einem lokalen rezeptiven Feld handelt. Die stärkeren Pfeile der ersten Schicht verdeutlichen, dass es sich hier eher um Referenzvektoren handelt, die auf typische Muster im Eingaberaum zeigen. Die zweite Schicht besteht im Prinzip aus einer Lage vollständig vernetzter Perzeptrons.

Betrachtet man ein Perzeptron mit nur einem einzigen verdeckten Neuron, wird die Aktivität des Ausgabeneurons zu:

$$y_k(\mathbf{x}) = w_k\, exp\left(-\frac{\|\mathbf{q}_j - \mathbf{x}\|^2}{\sigma_j^2}\right) + \theta_k \tag{2.13}$$

Damit sind die möglichen Entscheidungsregionen auf konzentrische Hypersphären um den Referenzvektor beschränkt. Fügt man mehrere Neuronen in die verdeckte Schicht ein, wird die effektive Entscheidungsregion eine beliebige Linearkombination aus solchen Hypersphären. Dabei ist zu beachten, dass der Beitrag eines einzelnen verdeckten Neurons sowohl erregend wie auch hemmend sein kann. Wesentlicher Faktor ist hier die Breite σ_j der einzelnen Aktivierungsfunktionen. Sie muss groß genug sein, damit benachbarte Neuronen effektiv interagieren können. Ist σ zu klein, wird die mögliche Entscheidungsregion einfach zu einer Summe isolierter Sphären.
Wählt man dagegen σ zu groß, geht der lokale Charakter der Gaußfunktionen verloren. Als heuristische Wahl bietet sich an, die Breiten proportional zum mittleren Abstand zu den nächsten Nachbarn zu setzen [Moody89]:

$$\sigma_j = \sigma_{fac} \cdot \bar{d}_{j,nn} \tag{2.14}$$

Zum Training eines RBF-Netzes werden zunächst die Referenzvektoren bzw. Stützstellen gesucht. Das weitere Vorgehen entspricht im wesentlichen dem Perzeptron-Algorithmus. Es wird also jeweils ein Trainingsmuster mit zugehörigem Klassenvektor präsentiert, die Aktivitäten der Neuronen in der verdeckten Schicht gemäß (2.10) berechnet und aus diesen wiederum die Ausgabeaktivitäten der Perzeptrons nach (2.11).
Die Gewichte $\boldsymbol{w}$ werden dann nach der Delta-Regel belernt, wobei der Gewichtsvektor natürlich nicht in Richtung des Eingabemusters, sondern in Richtung der Aktivitäten der verdeckten Schicht belernt wird:

$$\Delta \mathbf{w}_k(t) = \eta(t_k - y_k(\mathbf{x}))\boldsymbol{\Phi}(\mathbf{x}) = \eta(t) \cdot \delta \cdot \boldsymbol{\Phi}(\mathbf{x}) \tag{2.15}$$

Die Größe δ gibt dabei an, wie weit die Aktivität des entsprechenden Perzeptrons von der Soll-Ausgabe abweicht. Für die Lernrate werden stetig fallende Funktionen verwendet und z.B. nach einem exponentiellen Schema vermindert:

$$\eta(t) = \eta_0 \cdot \eta_{fac}^t \tag{2.16}$$

Für die Auswahl der Referenzvektoren für ein RBF-Netz, also die Zentren der verdeckten Schicht, gibt es drei verschiedene Vorgehensweisen.
Eine Möglichkeit besteht darin, die Referenzmuster einfach zufällig aus den Trainingsmustern auszuwählen. Eine weitere Möglichkeit ist, die Referenzvektoren durch selbstorganisierte Verfahren, beispielsweise eines GNG, erzeugen zu lassen. Dies ermöglicht die gezielte Platzierung von Referenzvektoren.
Dritte Alternative ist die der wachsenden RBF-Netze [Fritzke94]. Hier wächst ein GNG, dessen Einheiten mit einer zweiten Schicht an Perzeptrons vollständig vernetzt sind. Bei jedem Schritt werden sowohl die Zentren wie auch die Gewichte der Perzeptrons trainiert. Entscheidend ist, dass man hier den Klassifizierungsfehler direkt zur Aktualisierung der Ressourcevariablen verwenden kann. So werden dann neue Einheiten bevorzugt in solchen Regionen eingefügt, wo die Klassifizierung bisher schlecht ist.

Prinzipiell ist es auch möglich, die Lage der Zentren sowie die Breite der rezeptiven Felder in den Trainingsprozess einzubeziehen. Hier verwendet man eine Fehler- bzw. Kostenfunktion und deren Gradienten bezüglich der Perzeptron-Gewichte, der Referenzvektoren und der Breiten, um diese zu belernen.

2.3 Spezielle Lernregeln und Analysemethoden

Die Anwendungsgebiete neuronaler Netze hängen entscheidend von der verwendeten Lernregel ab. Für überwachtes Lernen eines Perzeptrons wird der Backpropagation-Algorithmus vorgestellt. Zur Kennzahlenextraktion und Dimensionsreduzierung hochdimensionaler Räume wird die Principal Component Analysis (PCA) verwendet. Verschiedene Algorithmen zur PCA werden vorgestellt und im Ergebnisteil mit dem vorhandenen Datenmaterial genauer untersucht. Die Independent Component Analysis (ICA) ist ein relativ neues Verfahren zur Datenanalyse und Separation statistisch unabhängiger Ausgabesignale.
Oftmals wird die Netzwerktopologie vor dem Training festgelegt und kann anschließend nicht mehr modifiziert werden. Die Netzwerkgröße hängt entscheidend von der Komplexität der Aufgabe bzw. des zu modellierenden Datenmaterials ab. Um eine Abschätzung der Netzwerkarchitektur und eine Voranalyse der Zeitreihen durchzuführen, werden im letzten Abschnitt dieses Kapitels Verfahren zur nichtlinearen Zeitreihenanalyse vorgestellt.

2.3.1 Backpropagation

Backpropagation-Netzwerke bestehen aus einer Eingabeschicht, einer oder mehreren verdeckten Schichten und einer Ausgabeschicht. Die Neuronen einer Schicht sind vollständig mit den Neuronen einer vorgelagerten bzw. nachfolgenden Schicht durch synaptische Verbindungen $\boldsymbol{w}$ verbunden. Die Synapsen können dabei sowohl verstärkend (positiv) als auch hemmend (negativ) wirken. Die Ausgabe des Neurons c_i berechnet sich aus der Aktivität a_i und einer differenzierbaren Ausgabefunktion $f(a)$ zu $o_i = f(a_i)$. Die Ausgaben $\boldsymbol{o}$ einer Schicht sind die Eingaben für die nachfolgende Schicht.

Backpropagation zählt zu den überwachten Lernverfahren. Die Trainingsdaten werden als Musterpaare aus Eingabevektor $\boldsymbol{x}$ und Zielwert t (Soll-Ausgabe) dem Netz präsentiert. Nach der Ermittlung des Ausgabefehlers E, werden dann die synaptischen Gewichte zwischen den Neuronen entsprechend ihrem Anteil am Gesamtfehler modifiziert. Die Berechnung der Ausgabe durch die Eingabedaten mit den aktuellen Gewichten von der Eingabeschicht zur Ausgabeschicht bezeichnet man als *Feedforward*. Das Anpassen der Gewichte von der Ausgabeschicht zur Eingabeschicht wird als *Feedbackward* bezeichnet.
Wird bei jedem Zyklus aus *Feedforward* und *Feedbackward* eine Anpassung der Gewichte vorgenommen, wird dies als *Online*-Lernen bezeichnet und ausschließlich in dieser Arbeit verwendet.

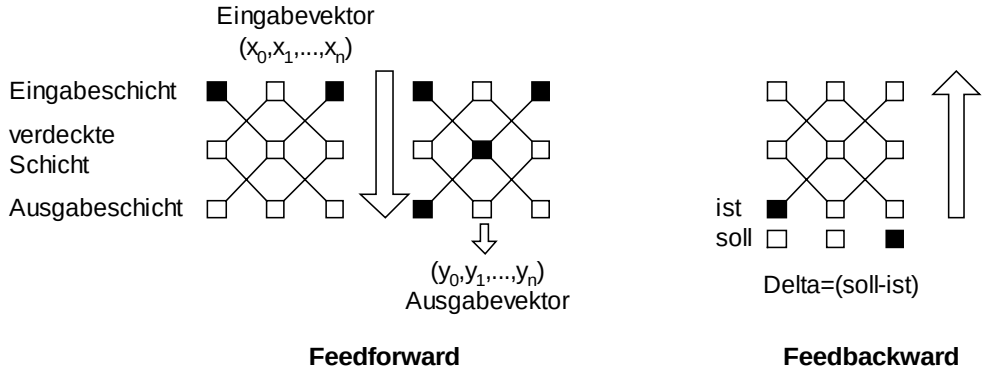

Abbildung 2.10 Lernzyklus bei einem Backpropagation-Netzwerk.

Feedforward

Bei diesem Schritt werden die Aktivitäten der Neuronen auf die Netzwerkeingabe berechnet. Die Aktivität a_j eines einzelnen Neurons c_j ist das Skalarprodukt von Eingabevektor $\boldsymbol{x}$ und Gewichtsvektor $\boldsymbol{w}_j$ des Neurons:

$$a_j = \mathbf{x}^T \cdot \mathbf{w}_j = \sum_i x_i w_{ij} \tag{2.17}$$

wobei i die Anzahl der Neuronen der vorgelagerten Schicht darstellt.

Die Ausgabe o_j des Neurons c_j wird mit der Ausgabefunktion $f(a_j)$ berechnet. Für $f(a_j)$ muss eine differenzierbare Funktion, wie die sigmoide Funktion, verwendet werden.

$$o_j = f(a_j) = \frac{1}{1+e^{-a_j}} \tag{2.18}$$

Diese Funktion begrenzt die Ausgabe o_j auf den Wertebereich [0;1].

Feedbackward

Das Anpassen der Gewichte geschieht durch das Gradienten-Abstiegs-Verfahren. Hierzu wird eine Fehlerfunktion E definiert.

$$E = \frac{1}{2}(t-y)^2 \tag{2.19}$$

Die partielle Ableitung dieser Funktion nach den Gewichten w_{ij} der einzelnen Neuronen c_j führt zu einem Ausdruck zur Ermittlung des Teilfehlers δ_j des Neurons zum Gesamtfehler.
Für Neuronen der Ausgabeschicht, mit der Netzwerkausgabe y_j und der Ableitung der sigmoiden Funktion $f'(a)$ gilt:

$$\delta_j = (t_j - y_j) \cdot f'(a_j) \tag{2.20}$$

$$f(a_j) = y_j \Rightarrow \delta_j = (t_j - y_j) y_j (1 - y_j) \tag{2.21}$$

Für Neuronen *k* der vorgelagerten, verdeckten Schicht gilt:

$$\delta_k = \sum_j w2_{ij} \delta_j \cdot f'(a_k) \tag{2.22}$$

$$f(a_k) = o_k \Rightarrow \delta_k = \sum_j w2_{ij} \delta_j \cdot o_k (1 - o_k) \tag{2.23}$$

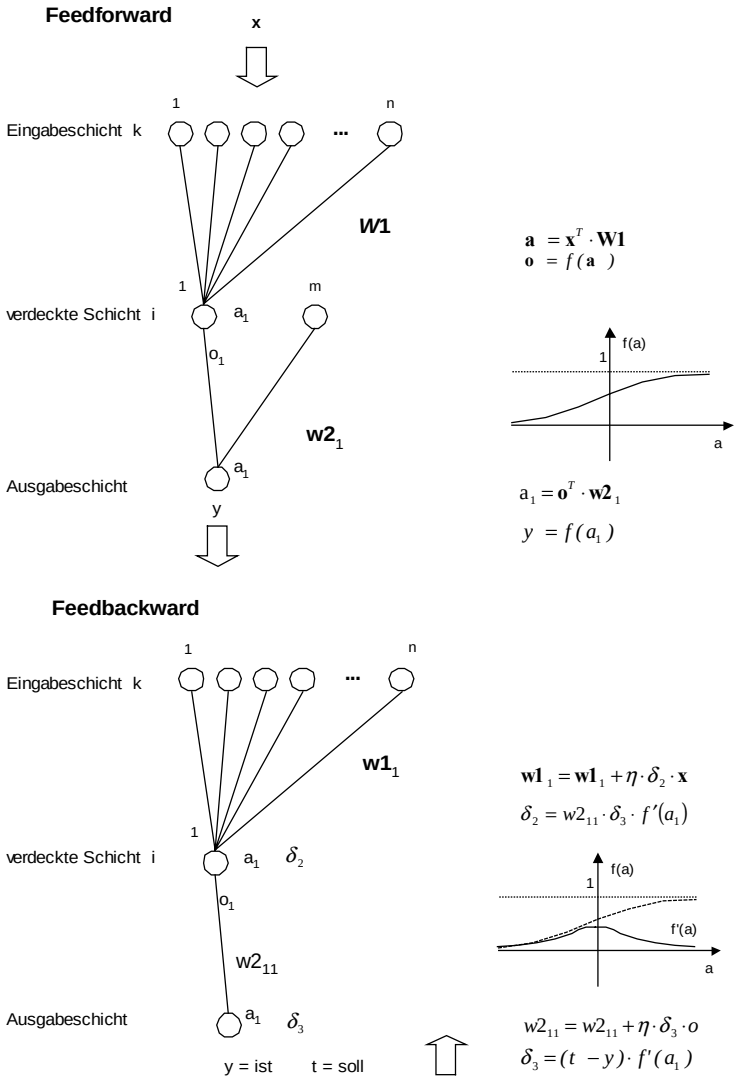

Abbildung 2.11: Schematische Darstellung von Feedforward und Feedbackward beim Backpropagation-Algorithmus bei einer verdeckten Schicht und einem Ausgabeneuron.

Die Gewichtsänderung Δw_{ij} eines Neurons c_j in Richtung des negativen Gradienten zur Minimierung des Gesamtfehlers ist wie folgt definiert:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \delta_j x_i \tag{2.24}$$

Der Faktor η ist die Lernrate und gibt an wie stark der Fehler δ in die Gewichtsänderung eingeht.

Während der Lernphase wechseln sich *Feedforward* und *Feedbackward* ständig ab. Bei jedem Zyklus wird ein Trainingsmusterpaar an das neuronale Netzwerk angelegt. Um zu verhindern, dass die Reihenfolge der Testmuster in die Gewichtsänderung mit eingeht, werden die Musterpaare zufällig aus den Trainingsdaten ausgewählt.
Abbildung 2.11 zeigt schematisch einen Zyklus aus *Feedforward* und *Feedbackward* für ein dreischichtiges neuronales Netzwerk, wie es auch für die Simulationen in Kapitel 3.2 realisiert wurde. Dargestellt ist nur ein Neuron der verdeckten Schicht. Das verwendete Netzwerk verfügt nur über ein einziges Ausgabeneuron zur Darstellung der Klasseneinteilung „*good*“ (Soll-Ausgabe t=0) bzw. „*bad*“ (Soll-Ausgabe t=1) der Trainingsmuster.

2.3.2 Principal Component Analysis (PCA)

Die Principal Component Analysis (PCA) oder Hauptkomponentenanalyse stellt eine klassische Methode zur Dimensionsreduzierung dar. Im wesentlichen wird eine orthogonale Transformation durchgeführt, die das alte Koordinatensystem $(x_1, x_2, ..., x_n)$ so in ein neues Koordinatensystem $(x_1', x_2', ..., x_n')$ überführt, dass die Streuung der Datensätze entlang der neuen Achsen maximal wird. Vergleichbar ist die PCA mit der bekannten Technik der Hauptachsentransformation z.B. von Trägheitsellipsoiden in der klassischen Mechanik. Die neuen Richtungen werden als Hauptkomponenten bezeichnet und entsprechen den Eigenvektoren der Kovarianzmatrix [Haykin99].
Die erste Hauptachse ist dabei so ausgezeichnet, dass sie die maximale Varianz liefert. Die weiteren Hauptachsen sind nach fallenden Varianzen geordnet, wobei das neue Koordinatensystem ein orthonormales Basissystem des Raums bildet. In Abbildung 2.12 ist ein Beispiel im zweidimensionalen Raum gezeigt. Man sieht, dass nach der Drehung des Koordinatensystems auf die neuen Achsen x' und y', die Projektion der Punkte auf die Achse x' maximale Varianz besitzt. Hier ist die x' die erste Hauptachse und y' die zweite Hauptachse.

Zur Dimensionsreduzierung kann man nun den transformierten Datensatz auf die Koordinaten höchster Varianz beschränken. Die zugrundeliegende Idee ist, dass statistisch relevante Informationen allein in der räumlichen Verteilung verborgen sind. Ist diese Voraussetzung erfüllt, liefert dieses Verfahren bei einer Dimensionsreduzierung den kleinstmöglichen Informationsverlust. Ist diese Voraussetzung nicht gegeben, stellt dieses Verfahren keine geeignete Methode zur Dimensionsreduzierung dar.

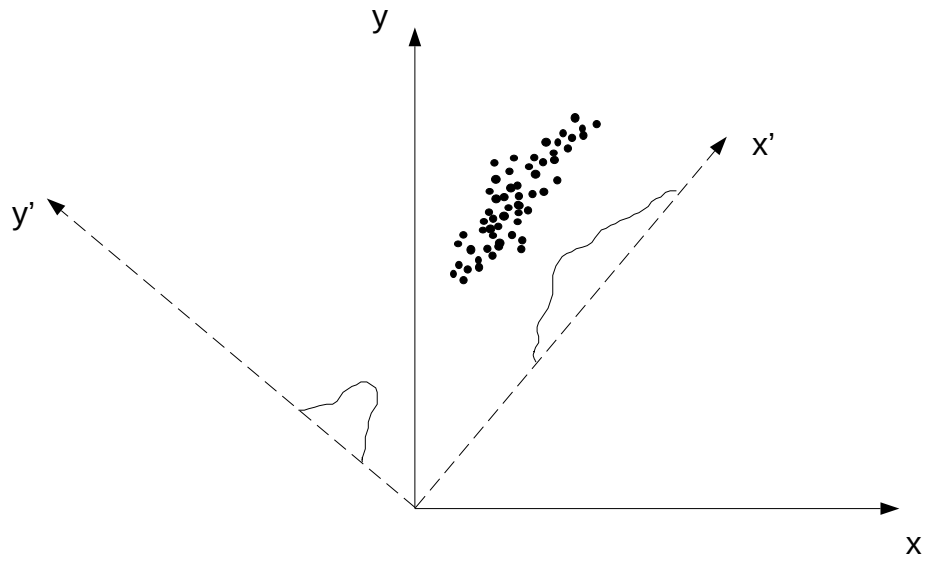

Abbildung 2.12: Drehung des Koordinatensystems auf die Hauptachsen unter Varianzmaximierung. Man sieht, dass die Projektion der Datenmenge auf die Achse x' maximale Varianz besitzt, d.h. es findet sich keine andere Achse, deren Projektion größere Varianz liefert. Die Achse x' ist also die erste Hauptachse.

Um maximale Information bei der Dimensionsreduzierung zu erreichen, sucht man nach einer linearen Transformation, die den mittleren quadratischen Fehler minimiert. Dies bedeutet, dass einige seiner Komponenten kleine Varianz besitzen und der Abstieg der Varianzen maximal wird.

Bei der PCA handelt es sich im wesentlichen um ein Eigenwertproblem, dessen nichttriviale Lösungen die Eigenwerte der Kovarianzmatrix $\boldsymbol{R}$ sind. Die Eigenwerte sind reell und positiv, was sich auf die Symmetrie der Kovarianzmatrix zurückführen lässt.

Seinen $\lambda_1, \ldots, \lambda_p$ die Eigenwerte von $\boldsymbol{R}$ und $\boldsymbol{u}_1, \ldots \boldsymbol{u}_2$ die Eigenvektoren von $\boldsymbol{R}$. Man erhält p Eigenwertgleichungen [Haykin99]:

$$\mathbf{R}\mathbf{u}_j = \lambda_j \mathbf{u}_j, \qquad j = 1, \ldots, p \tag{2.25}$$

Da die Eigenvektoren von $\boldsymbol{R}$ per Definition orthogonal sind, d.h.

$$\mathbf{u}_j^T \mathbf{u}_i = \delta_{ij}$$

folgt:

$$\mathbf{u}_j^T \mathbf{R}\mathbf{u}_j = \lambda_j \tag{2.26}$$

Sei $\boldsymbol{x}$ ein p-dimensionaler Vektor, der eine Datenmenge in einem p-dimensionalen Raum repräsentieren soll. Beim Abschneiden der Komponenten $m,\ldots,p$ macht man einen mittleren quadratischen Fehler:

$$e = \sum_{i=m}^{p} \sigma_i^2 = \sum_{i=m}^{p} \lambda_i \tag{2.27}$$

falls die Eigenwerte in fallender Reihenfolge geordnet sind. Der Fehler ist also gleich der Summe der Varianzen der Komponenten von ***x***, die bei der Dimensionsreduzierung vernachlässigt werden.
Die PCA kann damit durchgeführt werden, indem die zur Datenmenge gehörende Kovarianzmatrix diagonalisiert wird. Diese Methode wird im Folgenden als die „klassische“ genannt, da klassische numerische Verfahren zur Diagonalisierung Verwendung finden können [Press98].

Eine alternative Vorgehensweise zur klassischen Bestimmung der PCA ist die Verwendung eines korrelationsbasierten Hebb'schen Lernverfahrens. Der Unterschied ist, dass hier die Kovarianzmatrix weder berechnet noch diagonalisiert werden muss. Dies ist speziell dann von Vorteil, wenn die Zahl der zu berechnenden Hauptkomponenten sehr viel kleiner ist, als die Dimension des Eingabevektors.
Man betrachtet dazu ein einfaches Perzeptron, wie es in Kapitel 2.1.3 beschrieben wurde. Die Ausgabe für das lineare Neuron ist:

$$y = \sum_{i=1}^{p} w_i x_i =: \mathbf{w}^T \mathbf{x} \tag{2.28}$$

Für die Aktualisierung der Gewichte kann man folgende Hebb'sche Lernregel verwenden:

$$w_i(t+1) = w_i(t) + \eta y(t) x_i(t) \tag{2.29}$$

Die Lernregel ist korrelationsbasiert, da das präsynaptische und das postsynaptische Signal korreliert sind. Das Problem, dass die Gewichte hierbei divergieren, wird durch einen Normalisierungsterm behoben, der von Oja [O82] eingeführt wurde. Für eine kleine Lernrate η erhält man:

$$w_i(t+1) = w_i(t) + \eta y(t)[x_i(t) - y(t) w_i(t)] \tag{2.30}$$

Die Erweiterung um den nichtlinearen Term bewirkt, dass die Gewichte in einen stabilen Endzustand übergehen. Dieser Term wird in der Literatur als Decay-Term bezeichnet.
Diese Lernregel kann nun nach Sanger [Sanger89] in einer verallgemeinerten Form geschrieben werden:

$$\Delta w_{ji}(t) = \eta \left[y_i(t) x_i(t) - y_i(t) \sum_{k=1}^{j} w_{ki}(t) y_k(t) \right] \tag{2.31}$$

Es zeigt sich [Haykin99], dass diese Sanger-Lernregel in den Ausgabeneuronen die Hauptkomponenten und in den Gewichtsvektoren ***w*** die transformierten Hauptachsen ergeben, wobei diese nach fallenden Eigenwerten bzw. Varianzen geordnet sind. Dieser Algorithmus wird in der Literatur auch als GHA (Generalized Hebb Algorithm) bezeichnet.

Im Folgenden wird ein weiterer Algorithmus zur Durchführung der PCA beschrieben. Das besondere an dem APEX-Algorithmus (Adaptive Principal Components Extraction) ist, dass zur Laufzeit des Programms entschieden werden kann, wie viele Hauptkomponenten berechnet werden sollen.
Das benutzte Netzwerk ist in Abbildung 2.13 abgebildet. Die *Feedforward*-Verbindungen w_{mn} von der Eingabeschicht zur Ausgabeschicht werden nach der Hebb'schen Lernregel (2.29) belernt. Die lateralen Kopplungen a_{ji} innerhalb der Ausgabeschicht stellen die Rückkopplung des Netzes dar und werden anti-hebb'sch belernt.

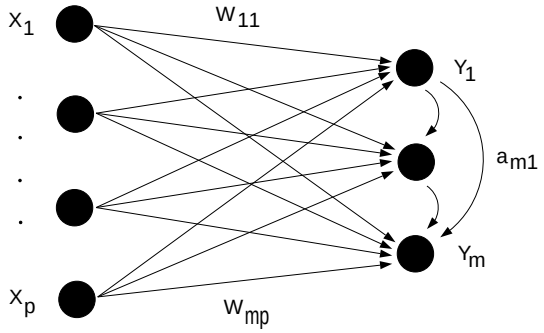

Abbildung 2.13: Vollständig vernetztes einschichtiges Netzwerk mit lateralen Kopplungen. Dabei entspricht x_j den Eingabeneuronen, y_i den Ausgabeneuronen, w_{ij} den synaptischen Kopplungen und a_{kl} den lateralen Kopplungen.

Sei $\mathbf{w}_m(t) = (w_{m1}(t),...,w_{mp}(t))^T$ der Gewichtsvektor und $\mathbf{a}_m(t) = (a_{m1}(t),...,a_{m,m-1}(t))^T$ der laterale Kopplungsvektor des Ausgabeneurons m, dann ist die Ausgabe dieses Neurons gegeben durch:

$$y_m(t) = \mathbf{w}_m^T(t)\mathbf{x}(t) + \mathbf{a}_m^T(t)\mathbf{y}_{m-1}(t) \qquad (2.32)$$

Es sei nun angenommen, dass die Ausgabeneuronen *1,. . . , m-1* des Netzwerkes bereits in einen stabilen Endzustand konvergiert sind, d.h.

$$\mathbf{w}_k(t) = \mathbf{q}_k, \qquad k = 1, \ldots, m\text{-}1$$
$$a_k(t) = 0, \qquad k = 1, \ldots, m\text{-}1$$

wobei $\boldsymbol{q}_k$ der k-te Eigenvektor der Kovarianzmatrix $\boldsymbol{R}$ ist, und die Eigenwerte bereits in fallender Reihenfolge geordnet sind:

$$\lambda_1 > \ldots > \lambda_{m-1}$$

Die Lernregeln werden definiert als:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) + \eta\left[y_m(t)\mathbf{x}(t) - y_m^2(t)\mathbf{w}_m(t)\right] \qquad (2.33)$$

$$\mathbf{a}_m(t+1) = \mathbf{a}_m(t) - \eta\left[y_m(t)\mathbf{y}_{m-1}(t) - y_m^2(t)\mathbf{a}_m(t)\right] \qquad (2.34)$$

Für die Lernrate η kann folgender optimaler Wert angegeben werden:

$$\eta_{m,opt(t)} = \frac{1}{\sigma_m^2(t)} = \frac{1}{\lambda_m(t)} \tag{2.35}$$

Der erste Term in der Klammer von Gleichung (2.33) stellt den hebb'schen Term dar, während der erste Term in der Klammer von Gleichung (2.34) den anti-hebb'schen Term repräsentiert. Die zweiten Terme in den Klammern garantieren jeweils die Konvergenz der Gewichte. Anti-Hebb'sch steht für das Minuszeichen vor der Klammer in Gleichung (2.34).
Durch Induktion kann gezeigt werden [Haykin99], dass, falls die Ausgabeneuronen 1, . . . , m-1 bereits konvergiert sind, das Ausgabeneuron m die m-te Hauptkomponente der Kovarianzmatrix extrahiert und die lateralen Kopplungen zwischen konvergierten Neuronen Null ist.
Die Konvergenz kann dabei im Einzelfall über die lateralen Kopplungen geprüft werden, die im Endzustand Null sein müssen.

2.3.3 Independent Component Analysis (ICA)

Die Independent Component Analysis (ICA) ist ein wichtiges statistisches Verfahren, mit dem Korrelationen höherer Ordnung in einer hochdimensionalen Datenmenge entfernt werden können. Im Vergleich dazu werden bei der PCA nur Korrelationen 2. Ordnung berücksichtigt. Ein Spezialfall der ICA ist die sogenannte Blind Source Separation (BSS). Die ersten Arbeiten über BSS wurden von Jutten et al. [Jutten91] veröffentlicht. Bis heute ist BSS und ICA Gegenstand vieler Forschungsprojekte.

Seien n unbekannte Quellensignale $s_i(t)$ mit $i = 1, \ldots, n$ zu jeder Zeit t statistisch unabhängig, d.h. die gemeinsame Wahrscheinlichkeitsdiche $p(\mathbf{s})$ ist gleich dem Produkt der marginalen Verteilungen:

$$p(\mathbf{s}) = \prod_{i=1}^{n} p_i(s_i) \tag{2.36}$$

Wenn nun lineare Mischungen dieser Quellensignale vorhanden sind, gelingt es der ICA, nur aus der Kenntnis der Mischungen die Quellensignale zu extrahieren. Die ICA wird verstärkt in den Bereichen wie Spracherkennung, medizinischer Signalverarbeitung und Bildverarbeitung eingesetzt.
Es gibt zahlreiche Algorithmen, die auf unterschiedliche Weise die ICA durchführen. So gibt es die nichtlinearen PCA-Algorithmen, siehe Oja et al. [Oja95] bzw. Karhunen [Karhunen96], die *marginale Negentropy* als Projektionsindex, um maximale Kurtosisprojektionen zu finden [Girolami97], *Maximum Likelihood Estimations* [Cardoso97], *Maximum Entropy* [Yang97] oder *Minimum Mutual Information* (MMI) [Yang97]. Die meisten dieser Algorithmen basieren auf neuronalen Netzen. In dieser Arbeit wird der MMI-Algorithmus verwendet, der die „gegenseitige Information" der Ausgabe minimiert.

Sei nun $\boldsymbol{A} \in \Re^{nxn}$ eine lineare Mischungsmatrix, d.h. die Anzahl der Quellen soll gleich der Anzahl der Mischungen sein, so gilt:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \tag{2.37}$$

mit

$$\mathbf{s}(t) = (s_1(t),...,s_n(t))^T$$
$$\mathbf{x}(t) = (x_1(t),...,x_n(t))^T$$

Die Komponenten $\boldsymbol{x}_i(t)$ von $\boldsymbol{x}$ sind lineare Mischungen der Quellen $s_i(t)$. Nur aus der Kenntnis von $\boldsymbol{x}(t)$ sollen die Quellensignale extrahiert werden.
Dazu betrachtet man folgende Transformation:

$$\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t) \tag{2.38}$$

$\boldsymbol{W}$ nennt man die Entmischungsmatrix, falls gilt:

$$\boldsymbol{W} = \boldsymbol{A}^{-1}$$
$$\Rightarrow \mathbf{y}(t) = \mathbf{W}\mathbf{A}\mathbf{s}(t) = \mathbf{s}(t) \tag{2.39}$$

Gleichung (2.38) ist allerdings nicht ganz exakt. Die Originalquellen können nur bis auf einen Skalierungsfaktor und beliebige Permutationen genau bestimmt werden. Dies liegt daran, dass man in Gleichung (2.39) ein Produkt zweier unbekannter Größen, nämlich $\boldsymbol{A}$ und $\boldsymbol{s}(t)$ hat.
Die Entmischungsmatrix schreibt sich korrekt:

$$\boldsymbol{W} = \boldsymbol{\Lambda}\boldsymbol{P}\boldsymbol{A}^{-1} \tag{2.40}$$

Wobei $\boldsymbol{\Lambda}$ eine Diagonalmatrix ist, die die Skalierungsfaktoren enthält, und $\boldsymbol{P}$ eine Permutationsmatrix, die die Quellen richtig permutiert. Permutationsmatrix heißt, dass in jeder Zeile und Spalte genau eine Eins steht. Ein Spezialfall ist die Einheitsmatrix.

Die Entmischungsmatrix $\boldsymbol{W}$ muss so gewählt sein, dass die Qutputkomponenten $y_a(t)$ von $\boldsymbol{y}(t)=(y_1(t), \ldots, y_n(t))^{\mathrm{T}}$ statistisch unabhängig werden, bzw. dass die statistischen Abhängigkeiten der Ausgabekomponenten untereinander minimiert werden. Diese Abhängigkeit kann man mit der Kullback-Leibler-Divergenz zwischen der gemeinsamen Wahrscheinlichkeitsdichtefunktion und dem Produkt der marginalen Wahrscheinlichkeits-dichtefunktionen der Ausgabe messen:

$$I(\mathbf{W}) = D[p(\mathbf{y};\mathbf{W}) \,\|\, \tilde{p}(\mathbf{y};\mathbf{W})] = \int p(\mathbf{y};\mathbf{W}) log \frac{p(\mathbf{y};\mathbf{W})}{\tilde{p}(\mathbf{y};\mathbf{W})} d\mathbf{y} \tag{2.41}$$

Mit der gemeinsamen Wahrscheinlichkeitsdichtefunktion $p(\boldsymbol{y};\boldsymbol{W})$ und dem Produkt der marginalen Wahrscheinlichkeiten $\tilde{p}(\mathbf{y};\mathbf{W})$:

$$\tilde{p}(\mathbf{y};\mathbf{W}) = \prod_{a=1}^{n} p_a(y_a;\mathbf{W}) \tag{2.42}$$

Falls die Komponenten y_a statistisch unabhängig sind, wird $I(\boldsymbol{W})$ zu Null.

Die Lernregel für $\boldsymbol{W}$ erhält man, indem man die Kullback-Leibler-Divergenz $I(\boldsymbol{W})$ mit einem Gradientenabstiegsverfahren minimiert:

$$\frac{d\mathbf{W}}{dt} = -\eta \frac{\partial I(\mathbf{W})}{\partial \mathbf{W}} \tag{2.43}$$

Gram-Charlier-Entwicklung

Um nun $I(\boldsymbol{W})$ berechnen zu können, benötigt man noch einen Ausdruck für die marginalen Wahrscheinlichkeiten $p_a(y_a;\boldsymbol{W})$ die nach Standardverteilungen entwickelt werden können. Allgemein gilt:

$$p_a(y_a) = \sum_{i=0}^{\infty} c_i K_i(y_a)\beta(y_a) \tag{2.44}$$

wobei c_i Entwicklungskoeffizienten, $K_i(y_a)$ orthogonale Polynome und $\beta(y_a)$ eine Standardverteilung darstellen. In dieser Arbeit werden zwei Entwicklungen, die Gram-Charlier und die Edgeworth-Entwicklung betrachtet.

Die Gram-Charlier-Entwicklung verwendet als Standardverteilung die Gaußverteilung:

$$\beta(y_a) = \alpha(y_a) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y_a^2}{2}} \tag{2.45}$$

und als orthogonale Polynome die Tschebyschev-Hermite-Polynome $H_i(y_a)$, die bei [Kendall69] definiert sind:

$$H_i(x)\alpha(x) = \left(-\frac{d}{dx}\right)^i \alpha(x) \tag{2.46}$$

Die ersten fünf Tschebyschev-Hermite-Polynome lauten:

$$H_o(x) = 1$$
$$H_1(x) = x$$
$$H_2(x) = x^2 - 1$$
$$H_3(x) = x^3 - 3x$$
$$H_4(x) = x^4 - 6x^2 + 3$$

und die Entwicklungskoeffizienten c_i, die man durch Ausnützen der Orthogonalitätsbeziehung der H_i erhält.

$$c_0 = 1$$
$$c_1 = 0$$

$$c_2 = \frac{1}{2}(\mu_2 - 1)$$
$$c_3 = \frac{1}{6}\mu_3$$
$$c_4 = \frac{1}{24}(\mu_4 - 6\mu_2 + 3)$$

wobei die μ_k die k-ten Momente von y_a sind:

$$\mu_k = E\left[y_a^k\right]$$

Die Wahrscheinlichkeitsdichtefunktion lässt sich damit schreiben als:

$$\begin{aligned} p_a(y_a) &= \sum_{i=0}^{\infty} c_i K_i(y_a)\beta(y_a) \\ &= \beta(y_a)(c_0 H_0 + c_1 H_1 + c_2 H_2 + c_3 H_3 + c_4 H_4 + ...) \\ &= \beta(y_a)\left(1 + \frac{1}{2}(\mu_2 - 1)H_2 + \frac{1}{6}\mu_3 H_3 + \frac{1}{24}(\mu_4 - 6\mu_2 + 3)H_4 + ...\right) \end{aligned} \quad (2.47)$$

Um die Rechnung zu vereinfachen, trifft man die Annahme, dass

$$\mu_2 = E\left[y_a^2\right] = 1 \qquad \forall a = 1,...,n \quad (2.48)$$

und für weitere Kumulanten 3. und 4. Ordnung von y_a definiert man:

$$\kappa_3^a = \mu_3^a = E\left[y_a^3\right] \quad (2.49)$$

$$\kappa_4^a = \mu_4^a - 3 = E\left[y_a^4\right] - 3 \quad (2.50)$$

mit der Skewness κ_3^a und der Kurtosis κ_4^a. Damit lässt sich die Wahrscheinlichkeitsdichtefunktion schreiben als:

$$\begin{aligned} p_a(y_a) &= \alpha(y_a)\left(1 + \frac{\kappa_3^a}{6}H_3(y_a) + \frac{\kappa_4^a}{24}H_4(y_a) + ...\right) \\ &\approx \alpha(y_a)\left(1 + \frac{\kappa_3^a}{6}H_3(y_a) + \frac{\kappa_4^a}{24}H_4(y_a)\right) \end{aligned} \quad (2.51)$$

Für die partielle Ableitung der Kullback-Leibler-Divergenz ergibt sich damit:

$$\frac{\partial I(\mathbf{W})}{\partial w_{ak}} \approx -\left(\mathbf{W}^{-T}\right)_{ak} + f\left(\kappa_3^a, \kappa_4^a\right)E\left[y_a^2 x_k\right] + g\left(\kappa_3^a, \kappa_4^a\right)E\left[y_a^3 x_k\right] \quad (2.52)$$

mit

$$f(y,z) = -\frac{1}{2}y + \frac{9}{4}z$$
$$g(y,z) = -\frac{1}{6}z + \frac{3}{2}y^2 + \frac{3}{4}z^2$$

Durch Ersetzen der Erwartungswerte E[.] durch die instantanen Werte ergibt sich in Matrixschreibweise:

$$\frac{d\mathbf{W}}{dt} = -\eta(t)\frac{\partial I(\mathbf{W})}{\partial \mathbf{W}} = \eta(t)\left(I - \left(\mathbf{f}\left(\kappa_3^a, \kappa_4^a\right)\mathbf{y}^2 - \mathbf{g}\left(\kappa_3^a, \kappa_4^a\right)y^3\right)\mathbf{y}^T\right)\mathbf{W}^{-T} \quad (2.53)$$

Bei jedem Aktualisierungsschritt müsste das Inverse einer Matrix berechnet werden, was viel Rechenzeit beanspruchen würde. Yang und Amari [Yang97] zeigen, dass folgende Substitution gilt:

$$\frac{d\mathbf{W}}{dt} \rightarrow \frac{d\mathbf{W}}{dt}W^T W \quad (2.54)$$

Mit dieser Beziehung kommt man zum natürlichen oder relativen Gradientenabstieg [Yang97], die auch für Koordinatensysteme, die nicht orthonormal sind, den steilsten Gradientenabstieg gewährleistet:

$$\frac{d\mathbf{W}}{dt} = \eta(t)\left(I - \phi_\kappa(\mathbf{y})\mathbf{y}^T\right)\mathbf{W} \quad (2.55)$$

mit

$$\phi_\kappa(\mathbf{y}) = \mathbf{f}(\kappa_3, \kappa_4) \circ \mathbf{y}^2 + \mathbf{g}(\kappa_3, \kappa_4) \circ \mathbf{y}^3$$

Gleichung (2.55) ist die fertige Lernregel, nach der die „*Mutual Information*“ der Ausgabe minimiert und somit statistische Unabhängigkeit zwischen den Komponenten des Ausgabevektors erzeugt wird.
$\phi_\kappa(y)$ bezeichnet man als die Nichtlinearität oder *Score-Function* der Lernregel.

Edgeworth-Entwicklung

Für die Edgeworth-Entwicklung betrachtet man die Fourier-Transformation von $H_r(x)\alpha(x)$. Wobei $H_r(x)$ Hermitsche-Polynome und $\alpha(x)$ wieder die Gauß’sche Normalverteilung darstellen [Kendall69].
Für die Wahrscheinlichkeitsdichtefunktion erhält man:

$$p_a(y_a) = \alpha(y_a)\left(1 + \frac{\kappa_3^a}{6}H_3(y_a) + \frac{\kappa_4^a}{24}H_4(y_a) + \frac{\left(\kappa_3^a\right)^2}{72}H_6(y_a) + \frac{\kappa_3^a\kappa_4^a}{6}H_7(y_a) + \ldots\right) \quad (2.56)$$

und für die Funktionen *f* und *g* ergeben sich:

$$f(y,z) = -\frac{1}{2}y - \frac{1}{2}y^3 + \frac{3}{4}yz \tag{2.57}$$

$$g(y,z) = -\frac{1}{6}z + \frac{1}{2}y^2 \tag{2.58}$$

Um diese Lernregeln implementieren zu können, müssen die Funktionen *f* und *g* definiert sein. Für die Gram-Charlier und die Edgeworth-Entwicklung müssen hierzu die Kurtosis und Skewness bestimmt werden, die mit (2.49) und (2.50) nach folgendem Schema bestimmt werden können:

$$\Delta\kappa_3^a = -\eta(t)\left(\kappa_3^a(t) - y_a^3\right) \tag{2.59}$$

$$\Delta\kappa_4^a = -\eta(t)\left(\kappa_4^a(t) - y_a^4 + 3\right) \tag{2.60}$$

Weitere Möglichkeiten für die *Score-Function* sind sogenannte *fixed algorithms*, bei denen eine feste Nichtlinearität $\phi_\kappa(\boldsymbol{y})$ verwendet wird.

$$\phi_\kappa(\mathbf{y}) = \left(f(y_1),\ldots,f(y_n)\right)^T \tag{2.61}$$

z.B. können folgende Funktionen eingesetzt werden [Yang97]:

$$\begin{aligned} f(y) &= y^3 \\ f(y) &= 2\tanh(y) \\ f(y) &= \frac{3}{4}y^{11} + \frac{15}{4}y^9 - \frac{14}{3}y^7 - \frac{29}{4}y^5 + \frac{29}{4}y^3 \end{aligned} \tag{2.62}$$

Um den Algorithmus zu testen, wurden fünf verschiedene Signale, wie Rechtecks-, Sägezahn-Sinus- und Rauschsignal, künstlich erzeugt. Unter der Annahme, dass die Signale statistisch unabhängig sind, wurden mittels einer Mischungsmatrix $\boldsymbol{A}$ fünf lineare Mischungen erzeugt. Mit Hilfe von Gleichung (2.48) und der Nichtlinearität (2.62) konnten die Quellensignale aus den Mischungen extrahiert werden. Das Ergebnis ist in Abbildung 2.14 zu sehen. Man sieht, dass bis auf Permutationen und Skalierungsfaktoren die Originalsignale mit sehr guter Genauigkeit rekonstruiert werden konnten.

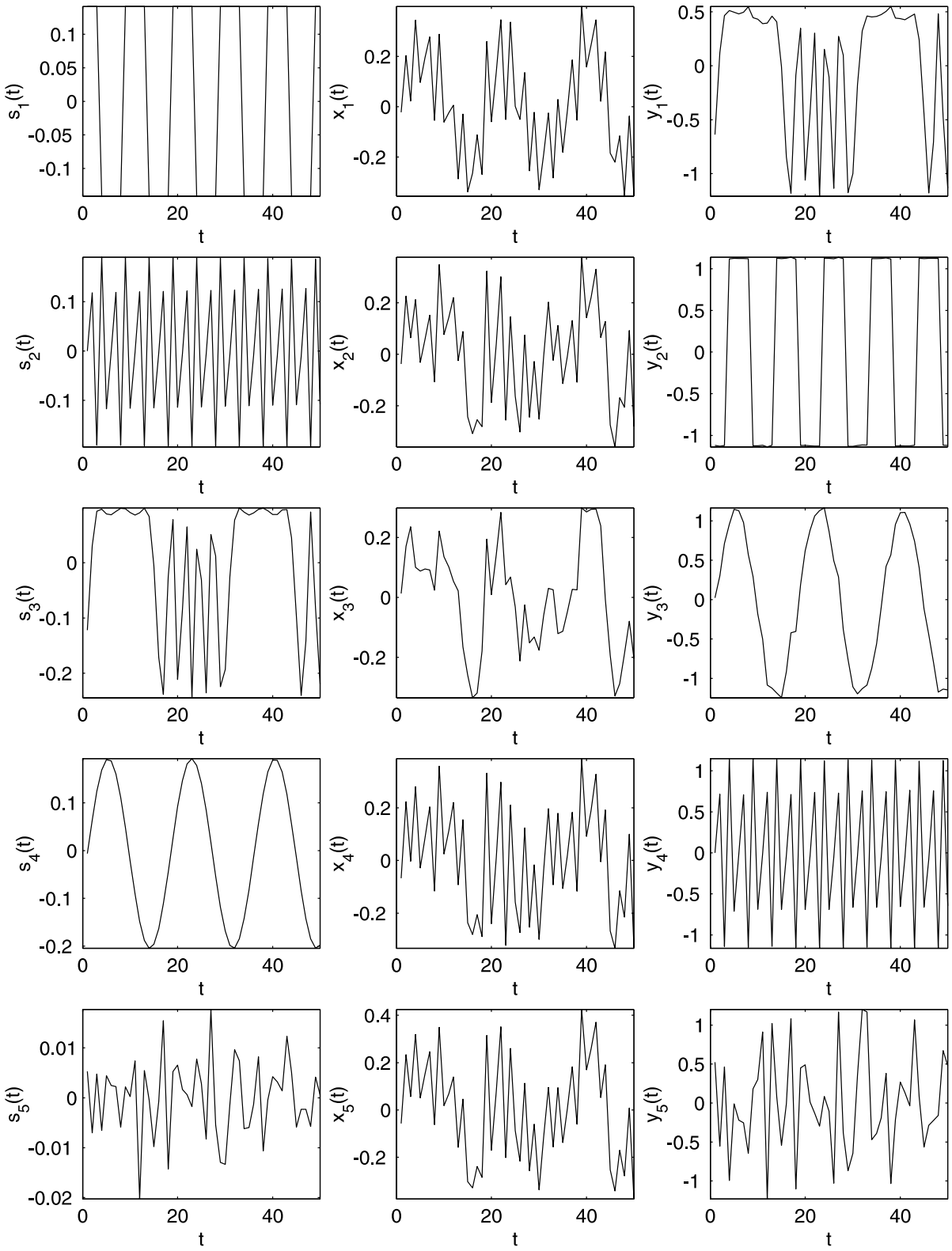

Abbildung 2.14: Beispiel für Blind Source Separation. In der linken Spalte sind die Quellsignale $s_i(t)$, in der Mitte die linearen Mischungen $x_i(t)$ und rechts die mit dem ICA Algorithmus extrahierten Quellsignale $y_i(t)$. Bis auf die Permutationen und Skalierungsfaktoren können die Quellsignale aus den linearen Mischungen extrahiert werden.

2.3.4 Nichtlineare Zeitreihenanalyse

Die Studie der nichtlinearen Dynamik und chaotischer Systeme trug wesentlich zum Verständnis vieler komplexer physikalischer Systeme bei und leistete einen wesentlichen Beitrag zur Entwicklung der nichtlinearen Zeitreihenanalyse [Weigend94]. Die komplexe Dynamik als nichtlineares Gleichungssystem aller Systemvariablen zu beschreiben ist dabei das formale Vorgehen. In Idealfällen lässt sich die Dynamik des Systems durch das Lösen dieser Gleichungssysteme untersuchen. Leider ist in vielen realen Situationen nicht genügend Information über alle Variablen, deren Interaktionen und Dynamiken verfügbar. In diesen Fällen ist oftmals keine Modellierung möglich, sondern es sind Zeitreihen einer oder mehrerer Systemvariablen verfügbar, die zur Rekonstruktion der Dynamiken verwendet werden können[Kulkarni97].

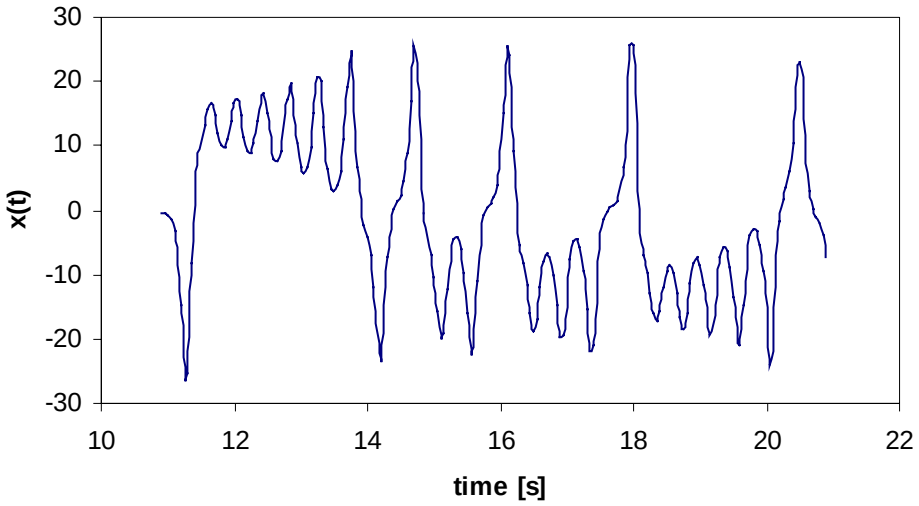

Abbildung 2.15: zeigt ein Beispiel einer deterministisch chaotischen Zeitreihe. Aufgetragen ist die Dimension x(t) des dreidimensionalen Lorenzgleichungssystems. Die verwendeten Parameter sind α=16.0, ρ=45.92, β=4.0. Als Δt wurde 0.001 sec zur besseren Genauigkeit benützt.

Bevor ein Modell, z.B. mit neuronalen Netzen, entwickelt werden kann, muss eine Datenanalyse durchgeführt werden mit dem Ziel, inhärente Informationen über die zugrundeliegende Natur der Daten zu finden. In den meisten Fällen zeigen sich während der Charakterisierung der Daten Informationen über das System:
linear – nichtlinear
deterministisch – stochastisch
regulär – chaotisch
Zu diesem Zweck wurden verschiedene Maßzahlen der Zeitreihe wie Autokorrelation, *Average Mutual Information* (AMI), Anzahl *false-nearest-neighbors*, Lyapunov Exponenten, eingebettete Dimension, fraktale Dimension etc. vorgeschlagen [Berthold99]. Zur quantitativen Charakterisierung der Zeitreihen werden im Folgenden die *Average Mutual*

Information als Verallgemeinerung der Autokorrelationsfunktion zur Bestimmung der Redundanz, und die eingebettete Dimension verwendet.

Average Mutual Information

Die *Average Mutual Information* (AMI) $S(\tau)$ ist ein Begriff aus der Informationstheorie, der zur Quantifizierung der Redundanz einer Zeitreihe verwendet werden kann. Als Erweiterung zur Autokorrelationsfunktion berücksichtigt die AMI auch Nichtlinearitäten.

$$S(\tau) = \sum_{i,j} p_{ij}(\tau) \log_2 p_{ij}(\tau) - 2\sum_{i} p_i \log_2 p_i \qquad (2.63)$$

Die AMI gibt die mittlere Information über s_n an, wenn $s_{n-\tau}$ bekannt ist. Ist s_n statistisch unabhängig von $s_{n-\tau}$, dann ist $S(\tau)=0$. Für $\tau=0$ gibt die AMI die Information an, die durch die Messung von s_n gewonnen wird und entspricht der Shannon-Entropie.
Je näher der Wert der AMI an der Shannon-Entropie liegt, desto stärker korreliert bzw. redundant sind die Datenpunkte.
Abhängig ist die AMI vom Informationsgehalt der Zeitreihe und kann als Maß für die Chaotik der Zeitreihe verwendet werden. Je chaotischer die Zeitreihe, desto größer ist die durch die Messung gewonnene Information. Eine weitere Abhängigkeit ergibt sich durch das Samplingintervall. Eine großes Samplingintervall führt zu einer weniger starken Redundanz aufeinanderfolgender Datenpunkte und damit zu einem größerem Informationsgewinn durch die Messung.

Einbettende Dimension und Vorhersage

Das Verhalten eines dissipativen Systems wird von m Zustandgrößen bestimmt. Der Verlauf kann im Fall deterministischer Systeme durch einen m-dimensionalen Attraktor beschrieben werden. Der Attraktor ist eine, abhängig von der Chaotik des Problems, mehr oder minder komplexe Trajektorie im Phasenraum, gegen die das System konvergiert.
Dissipative Strukturen sind dabei dynamische Ordnungszustände, die in Form räumlicher Muster mit stehenden Wellen verglichen werden können und denen zu ihrer Aufrechterhaltung ständig Energie zugeführt werden muss.

Oftmals ist allerdings nur die Messung einer Zustandsvariablen als Funktion der Zeit gegeben, aus der dann die zeitliche Entwicklung des m-dimensionalen Systems zu bestimmen ist. Unter bestimmten Voraussetzungen und Annahmen ist dieses Vorgehen gerechtfertigt.

Eine Annahme ist, dass die zeitliche Entwicklung der gemessenen Observablen abhängig ist, von allen m den Prozess bestimmenden Größen. Diese Annahme setzt starke Kreuzkorrelationen der Parameter voraus, die zu einer starken Reduktion der Freiheitsgrade und damit zu einer niedrigen Dimensionalität des Attraktors führt [Haken83].

Der Einfluss aller Zustandsgrößen auf die zeitliche Entwicklung der gemessenen Größe kann auch als Informationsfluss von den nicht-beobachteten zu den beobachteten Variablen betrachtet werden. In der Literatur werden zur Verdeutlichung häufig die Lorenzgleichungen verwendet.

$$\frac{dx}{dt} = \sigma(y - x) \tag{2.64}$$

$$\frac{dy}{dt} = \rho x - y - xz \tag{2.65}$$

$$\frac{dz}{dt} = -\beta z + xy \tag{2.66}$$

Abhängig von der Kopplungsstärke zwischen den einzelnen Variablen können evtl. nicht alle Freiheitsgrade aus einer Zeitreihe rekonstruiert werden. Wird z.B. nur die Variable *x* gemessen, ist die Information über *z* vom Informationsfluss über *y* abhängig, da $\dot{x}$ nicht direkt von *z* abhängig ist. Für die Variable *y* ist die Kopplung zu *z* über den Term *xz* gegeben. Ist *x*=0, so bewirkt eine große Änderung in *z* nur eine kleine Änderung in *y* und damit auch nur eine kleine Änderung in *x*.
Dies hat zur Folge, dass während der Bestimmung von *z* aus der Größe *x* sich ein kleiner Fehler der *x*-Messung, aufgrund der stark gekoppelten Gleichungen, stark auf die Genauigkeit von *z* auswirkt. Dieses Phänomen wird Rauschverstärkung genannt und führt in diesem Fall dazu, dass nicht alle Freiheitsgrade des Systems aus einer Zeitreihe rekonstruiert werden können.

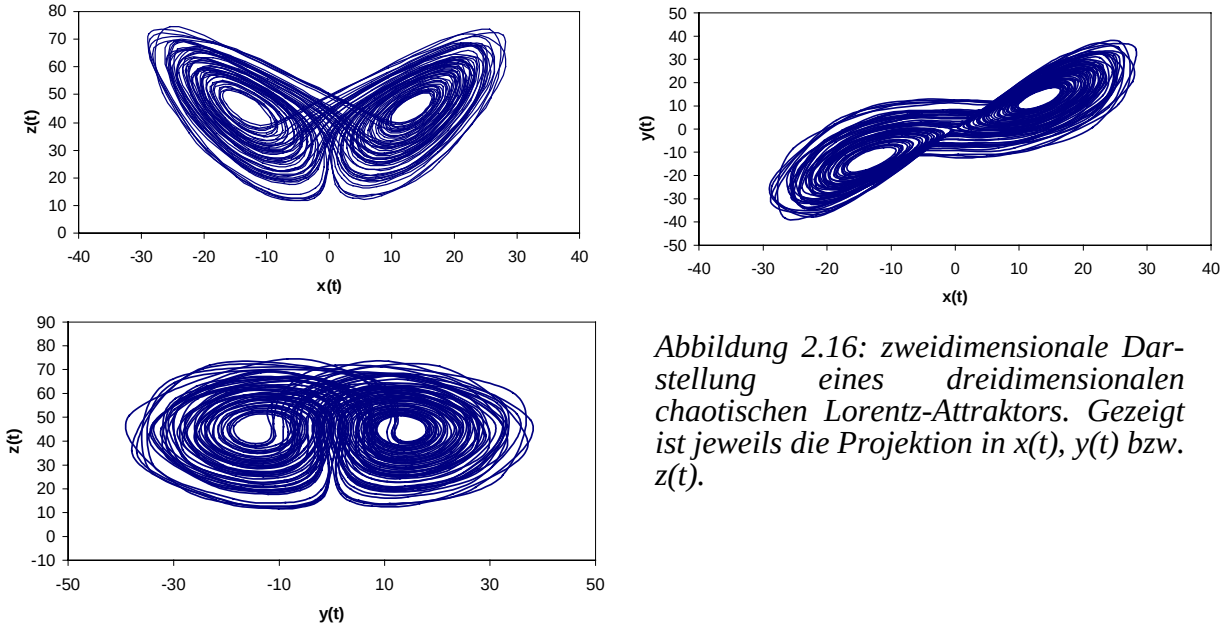

Abbildung 2.16: zweidimensionale Darstellung eines dreidimensionalen chaotischen Lorentz-Attraktors. Gezeigt ist jeweils die Projektion in x(t), y(t) bzw. z(t).

Die skalare Messung der Größe *x* entspricht einer Projektion eines m-dimensionalen Zustandes auf eine seiner Phasenraumachsen, wobei Information verloren geht. Diese kann teilweise durch Einbettungsmethoden rekonstruiert werden. Ist der dadurch verursachte Fehler größer als die Standardabweichung der Messreihe, entspricht dies einer Rekonstruktions-

Rauschverstärkung. Das rekonstruierte System erscheint um so weniger determiniert, je weniger Information aus den skalaren Messungen durch die Rekonstruktion erhalten wird.

Für den Großteil aller nichtlinearen Methoden wird zuerst eine Rekonstruktion des Attraktors nötig. Ist sie nicht zuverlässig, sind auch alle Ergebnisse dieser Methoden unzuverlässig.
Die Attraktor-Rekonstruktion muss gewährleisten, dass der m-dimensionale Originalattraktor des Systems und die in einem dim_E-dimensionalen Rekonstruktions-Zustandsraum eingebettete skalare Zeitreihe topologisch äquivalent ist. Dies bedeutet, dass jede Umgebung eines Punktes auf die Umgebung seines Bildpunktes bijektiv abgebildet wird. Nur dann hat sowohl der originale als auch der eingebettet Attraktor gleiche Dimension und gleiche Lyapunovexponenten.

Eine Methode zur Bestimmung der Attraktordimension stellt die Verzögerungseinbettung dar. Im Fall von niederdimensionalem Chaos wird diese Methode durch das Einbettungstheorem von F. Takens [Takens97] mathematisch gestützt und stellt eine Erweiterung des allgemeinen Einbettungstheorems von H. Whitney [Whitney36] dar.

Takens Theorem:
Sei M eine m-dimensionale kompakte Mannigfaltigkeit.
Für Paare (F,ν), wobei F ein glattes Vektorfeld und ν eine glatte Funktion auf M ist, ist es eine generische Eigenschaft, dass

$$\Phi_{F,\nu}(y): M \to R^{2m+1},$$

definiert durch

$$\Phi_{F,\nu}(y) = (\nu(y), \nu(\varphi_1(y)), \ldots, \nu(\varphi_{2m}(y)))^T$$

mit

φ_i *Fluss von F*

eine ausreichende Einbettung ist.

Das Theorem besagt, dass für eine Einbettungsdimension $dim_E \geq 2m+1$ der Attraktor im Rekonstruktionsraum voll entfaltet, also mit dem Originalattraktor topologisch äquivalent ist.

Generische Eigenschaft heißt, dass es isolierte Fälle gibt, für die diese Eigenschaft nicht zutrifft, dies aber durch eine kleine Störung in Form einer Parameteränderung geändert werden kann. Zum Beispiel, wenn eine Sinuswelle genau mit deren Periode T bzw. T+ε gesampelt wird.

Konkret erfolgt die Verzögerungseinbettung, indem man dim_E-dimensionale Verzögerungsvektoren bildet, deren aufeinanderfolgende Komponenten um τ_E verzögerte Zeitreihenwerte sind, also $[x(n), x(n+\tau_E), x(n+2\tau_E), \ldots, x(n+(dim_E-1)\tau_E]$.

Wichtige Parameter sind die Einbettungsdimension dim_E und die Einbettungsverzögerung τ_E.

Bei ausreichender Einbettungsdimension dim_E ist mit der Verzögerungseinbettung die Eindeutigkeit des Systemzustandes erfüllt. Das bedeutet, dass die dim_E Messwerte einer Zustandsgröße mit dem zeitlichen Abstand τ_E dazu ausreichen, einen Systemzustand

eindeutig festzulegen. Bei deterministischen Systemen stellt der Verzögerungsvektor eindeutig genau einen Systemzustand dar. Dies führt dazu, dass sich die Trajektorien im Phasenraum nicht schneiden und zukünftige Zustände eindeutig durch die Zustände in der Vergangenheit bestimmt sind.

In Takens Theorem wird eine Einbettungsdimension $dim_E=2m+1$ gefordert, wobei m die Dimension des Systemattraktors ist. In realen Situationen ist weder m *a-priori* bekannt, noch ist die Vorrausetzung von Takens Theorem einer unendlich langen, rauschfreien Zeitreihe mit unendlich hoher Samplingrate gegeben. Theoretisch wäre die Einbettung unabhängig von der Einbettungsverzögerung τ_E. In realen Situationen hängt die topologische Äquivalenz des rekonstruierten mit dem Originalattraktor entscheidend von τ_E ab.

Einbettungsverzögerung τ_E

Die optimale Einbettungsverzögerung τ_E ist abhängig von der jeweiligen Anwendung. Zur Bestimmung von τ_E wurden einige *ad-hoc*-Algorithmen vorgeschlagen.
Einfachste Möglichkeit besteht im einfachen Ausprobieren unterschiedlicher Einbettungsverzögerungen τ_E. In Abbildung 2.17 wird eine passende und eine zu hohe Einbettungsverzögerung gezeigt. Abbildung 2.17a zeigt eine gute Entfaltung des Attraktors und lässt auf eine akzeptable Einbettungsverzögerung schließen. Diese Methode ist natürlich sehr subjektiv und lässt sich für Dimensionen > 3 nicht mehr anwenden.

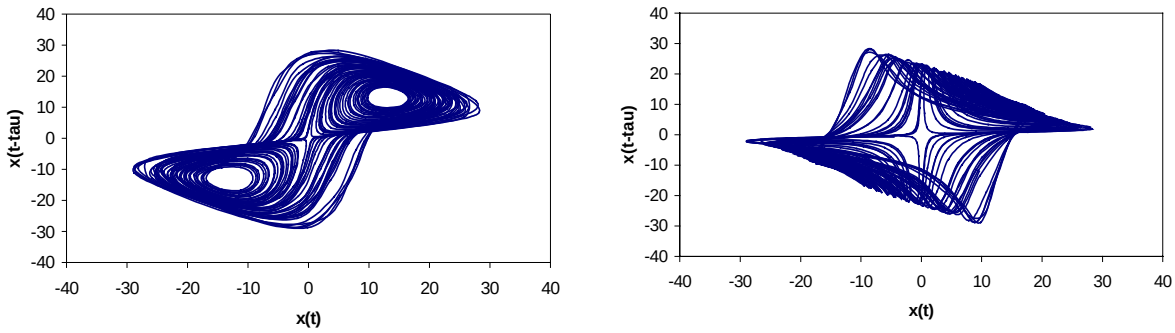

Abbildung 2.17: zweidimensionale Projektion des rekonstruierten Lorentzattraktors aus x(t) für a) τ_E=0,1 sec und b) τ_E=0,2 sec. Die gute Entfaltung in Bild a) lässt auf eine passende Einbettungsverzögerung von τ_E=0,1 sec schließen.

Als eine qualitative Methode zur Bestimmung der Einbettungsverzögerung wird die Verwendung der *Average Mutual Information* vorgeschlagen [Fraser86]. Bei der Wahl der Einbettungsverzögerung sollen aufeinanderfolgende Vektorkomponenten weder zu stark korreliert, noch total unkorreliert sein. Die erste und letzte Vektorkomponente müssen noch eine Korrelation aufweisen. Die Verwendung einer passenden Einbettungsverzögerung kann auch als nachträgliche Korrektur einer nicht optimalen Samplingrate betrachtet werden.

Als akzeptable Einbettungsverzögerung τ_E wählt man den Wert, bei dem die *Average Mutual Information S(n)* das erste Minimum erreicht. Falls kein Minimum vorhanden ist, wählt man $\tau_E = 1$.

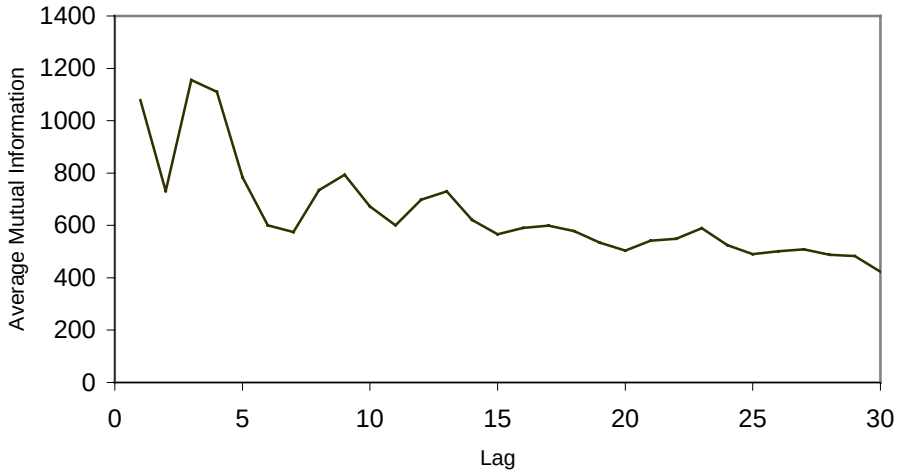

Abbildung 2.18: Average Mutual Information der Funktion x(t) der Lorenzgleichungen. Das erste Minimum indiziert eine passende Einbettungsverzögerung τ_E. In dieser Zeitreihe wird das erste Minimum bei einem Lag von 2 erreicht, was einer Verzögerung von 0,1 sec entspricht.

Zur Überprüfung dieses Verfahrens wurde die *Average Mutual Information* mit verschiedenen Zeitverzögerungen berechnet und in Abbildung 2.18 aufgetragen. Als Parameter für die Lorenzgleichungen wurden $\alpha=16.0$, $\rho=45.92$, $\beta=4.0$ und für Δt wurde 0.001 sec verwendet. Jeder 50. Wert wurde zur Visualisierung und Berechnung verwendet. Das erste Minimum in Abbildung 2.18 erscheint bei einem Lag von 2, entspricht damit einer Einbettungsverzögerung von 0,1 sec. und stimmt mit der subjektiven Bestimmung einer passenden Einbettungsverzögerung in Abbildung 2.17 überein.

Einbettungsdimension

Als passende Einbettungsdimension für den rekonstruierten Attraktor gibt Takens Theorem die Gleichung $dim_E=2m+1$ an. Die Attraktordimension m ist dabei normalerweise nicht bekannt. Die Dimensionen $dim_E>m$ des Einbettungsraumes sind dabei nur vom Rauschanteil des Signals besetzt und geben üblicherweise keine Information über die eigentliche Dynamik des Systems. Um unnötigen Rechenaufwand zu vermeiden, soll die minimale Einbettungsdimension ermittelt werden, bei der die topologische Äquivalenz noch gewährleistet ist. Eine Möglichkeit, die minimale Einbettungsdimension zu bestimmen, ist, die Entfaltung des Attraktors im Phasenraum zu betrachten. Falls der rekonstruierte Attraktor so weit entfaltet ist, dass sich die Trajektorien nicht mehr schneiden und somit jeder Punkt eindeutig bestimmt ist, kann von einer ausreichenden Einbettungsdimension ausgegangen werden.
Eine oftmals zitierte Methode, die minimale Einbettungsdimension zu bestimmen, stellt dabei die von Kennel et al. in [Kennel92] vorgestellte Methode der „*false-nearest-neighbors*“ (fnn).
Erfolgt die Attraktor-Rekonstruktion in einem Raum, dessen Dimension kleiner ist, als die dem System zugrundeliegenden Attraktor, liegen Punkte nicht aufgrund der bestimmenden Dynamik nebeneinander, sondern aufgrund der Projektion des Attraktors auf einen zu

niedrigdimensionalen Raum. Phasenraumpunkte werden durch die Rekonstruktion in Umgebungen falscher Nachbarn projiziert.

Die Vorgehensweise des fnn-Algorithmus ist, die Einbettungsdimension so lange zu erhöhen, bis keine falschen Nachbarn mehr vorhanden sind. Um falsche Nachbarn zu ermitteln, die das Resultat einer unzureichenden Auswahl der Dimensionsgröße dim_E sind, verwenden wir den Vektor

$$s(n)= [x(n), x(n+\tau_E), x(n+2\tau_E),...,x(n+(dim_E-1)\tau_E)].$$

Für eine gegebene Dimension dim_E hat dieser Vektor einen nächsten Nachbarn, s_{NN}, wobei die Nähe auf dem euklidischen Abstand basiert. Wenn dieser Abstand für die Dimension dim_E klein ist, aber relativ groß, wenn er für die Dimension dim_E+1 berechnet wird, können wir annehmen, dass der kleinere Abstand zum Nachbarn aufgrund einer Projektion von einem höher-dimensionalen Attraktor auf eine kleinere Dimension entstanden ist. Wenn unter der Verwendung der Dimension dim_E+1 die beiden Punkte voneinander de-projiziert werden, handelt es sich wirklich um falsche Nachbarn.

Für den Algorithmus zur Bestimmung der kleinsten Dimension bestimmt man für eine kleine Dimension den nächsten Nachbarn. Dieser Abstand wird mit dem Abstand für dim_E+1 verglichen usw. bis ein deutlicher Anstieg des Abstands auftritt. Ist die Änderung dieses Abstandes größer als ein heuristischer Schwellwert R_{tol}, so handelt es sich um einen falschen Nachbarn, d.h. falls gilt

$$\frac{\left\|\mathbf{x}_i^{dim_E+1} - \mathbf{x}_j^{dim_E+1}\right\| - \left\|\mathbf{x}_i^{dim_E} - \mathbf{x}_j^{dim_E}\right\|}{\left\|\mathbf{x}_i^{dim_E} - \mathbf{x}_j^{dim_E}\right\|} > R_{tol} \qquad (2.67)$$

Für jeden Verzögerungsvektor $\boldsymbol{x}_i$ der Zeitreihe im dim_E-dimensionalen Rekonstruktionsraum wird dessen nächster Nachbar $\boldsymbol{x}_j$ gesucht und deren euklidische Distanz berechnet. Dann wird die Dimension um Eins erhöht, was bei der Verzögerungseinbettung der Berücksichtigung eines weiteren um τ_E verzögerten Zeitreihenwertes entspricht, und berechnet deren Abstand.
Ist die relative Änderung zum ursprünglichen Abstand größer als ein heuristischer Wert R_{tol} so handelt es sich um falsche Nachbarn.
Falls der Anteil der Punkte des Attraktors, für die (2.67) gilt, gleich Null ist oder zumindest verschwindend gering, sind keine oder zumindest nur noch wenige falsche Nachbarn vorhanden und die Einbettungsdimension ist ausreichend hoch gewählt.
Die Darstellung der eingebetteten Dimension gegen die Anzahl der falschen nächsten Nachbarn gibt uns darüber hinaus noch weitere Informationen. Falls die Kurve den Nullwert erreicht und auch bei höheren Dimensionen Null bleibt, kann von einer deterministischen Dynamik des Systems ausgegangen werden. Falls die Kurve für höhere Dimensionen jedoch ansteigt, sind bis zu einem gewissen Maß stochastische oder zumindest farbige Rauschanteile enthalten. Als farbiges Rauschen wird dabei korreliertes, frequenzabhängiges Rauschen bezeichnet, das Wechselbeziehungen zwischen Punkten, die einen kleinen zeitlichen Abstand voneinander haben, besitzt. Falls die Anzahl der fnn's nie den Nullwert erreichen, ist die

zugrundeliegende Dynamik wahrscheinlich überwiegend durch stochastische Anteile bestimmt.

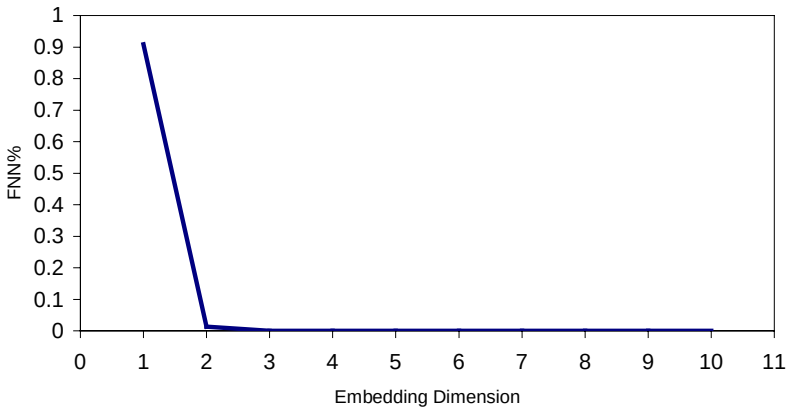

Abbildung 2.19: prozentualer Anteil falscher Nachbarn bei der x(t) Zeitreihe der Lorenzgleichungen. Bei einer einbettenden Dimension von drei fällt der Anteil der falschen Nachbarn auf Null. Dieses Beispiel zeigt, dass die fnn-Methode zumindest bei den Lorenzgleichungen funktioniert.

Die Werte für R_{tol} sollten in der Praxis zwischen $10 \geq R_{tol} \geq 50$ liegen [Abarbanel93]. Dieses Kriterium funktioniert in einigen Fällen, ist aber leider nicht völlig korrekt und kann als ad-hoc-Ansatz verwendet werden. Bei der Ermittlung eines Abstandsschwellenwerts, mit dem eine einbettende Dimension festgelegt werden soll, sollte man beachten, dass die Nähe relativ ist, weil die Distanzskala relativ zur Dimension dim_E und zur Größe des Attraktors ist. Der Schwellenwert sollte deshalb nicht nur auf der euklidischen Distanz, sondern auch auf der Größe des Attraktors basieren. In [Patterson96] schlägt DW Patterson vor, die Attraktorgröße durch die mittlere quadrierte Variabilität der Beobachtungen abzuschätzen und das Kriterium für falschen Nachbarn folgendermaßen zu definieren.

Mit der Abschätzung der Attraktorgröße R_A:

$$R_A^2 = \frac{1}{N}\sum_{k=1}^{N}(s(t)-\bar{s}) \quad \text{wobei gilt} \qquad \bar{s} = \frac{1}{N}\sum_{k=1}^{N}s(k) \tag{2.68}$$

und R_d^2 als quadrierten euklidischen Abstand zwischen $s(n)$ und s_{NN}

wird das vorgeschlagene Kriterium für falsche Nachbarn modifiziert, so dass, wenn gilt:

$$\frac{R_{d+1}(k)}{R_A} \geq 2 \tag{2.69}$$

die $s(k)$ und s_{NN} (beide mit der Dimension dim_E) als falsche Nachbarn deklariert werden. Die für die Einbettung erforderliche Dimension ist also dim_E+1.

Werden diese Methoden auf den Lorenzattraktor angewendet, fällt die Anzahl der falschen Nachbarn für den Wert von dim_E=3 auf Null, wodurch gezeigt wird, dass diese Vorgehensweise in der Praxis durchaus angewendet werden kann.

3. Simulationen und Ergebnisse

Aktuelle Methoden der Datenanalyse werden im Folgenden auf verschiedene Datensätze realer prozessierter Si-Wafer angewandt. Im wesentlichen handelt es sich hierbei um nichtlineare multisensorische Zeitreihen, die im ersten Kapitel mit unüberwachten Methoden, im zweiten Kapitel mit überwachten Methoden der Datenanalyse untersucht werden. Vorrangiges Ziel dieser Datenanalyse sind Informationsgewinnung aus den hochdimensionalen Datenräumen, Dimensionsreduzierung und Kennzahlenextraktion zur anschließenden Klassifikation und Detektion von abnormalen Prozessverläufen.
Das dritte Kapitel behandelt die Mustererkennung und Klassifikation hochdimensionaler Testdatenfelder mit überwachten und unüberwachten Methoden.

3.1 Unüberwachte Methoden zur Analyse nichtlinearer multisensorischer Datensätze

Die in diesem Kapitel verwendeten Zeitreihen entstammen Prozessanlagen der Trockenätzung und wurden während der Prozessierung einzelner Si-Wafer aufgezeichnet. Wie in der Einleitung kurz erwähnt, werden Kontrollmessungen des Prozessergebnisses in der Regel stichprobenartig durchgeführt. Dies führt im ungünstigsten Fall dazu, dass Scheiben mit niedriger Ausbeute nicht sofort erkannt und verworfen, sondern oftmals erst nach Durchlaufen der kompletten Fertigung detektiert werden. Hinzu kommt, dass mit aufwendigen Analysen die Problemanlage erst ausfindig gemacht werden muss, die dann häufig bereits viele Lose (fehl-)prozessiert hat.
Um dem Ziel, Abnormitäten des Prozesses *in situ* zu detektieren und damit eine höhere Produktivität und Produktqualität zu gewährleisten, einen Schritt näher zu kommen, werden diese multisensorischen Zeitreihen mit neuronalen Netzwerken analysiert. Da eine Korrelation zu den Ausbeutedaten zur Zeit aufgrund der fehlenden Datenbasis nicht möglich ist und außerdem auch infolge der vielen Prozessschritte, die einen Einfluss auf die Ausbeute haben, nicht erfolgversprechend erscheint, liegt keine *a-priori*-Information über das Prozessergebnis vor. In diesem Kapitel werden deshalb unüberwachte Methoden der Datenanalyse verwendet.

Speziell wurden die PCA als Verfahren zur Dimensionsreduzierung unter Berücksichtigung maximaler Varianz, ICA zur Bestimmung statistisch unabhängiger Vektoren und Kohonennetze zur Klassifikation, untersucht.

3.1.1 Datenvorverarbeitung

Typische Signalverläufe sind in Abbildung 3.1 dargestellt. Die erste Schwierigkeit der Vorverarbeitung besteht bereits darin, dass Prozessanlagen potentiell mehrere hundert Variablen liefern könnten. Die Schnittstelle zu den untersuchten Anlagen ist auf 9600 Bd begrenzt. Um wichtige Variablen noch mit einer vernünftigen Samplingrate abtasten zu können, muss bereits im Vorfeld eine Parameterselektion wahrscheinlich wichtiger Variablen vorgenommen werden. Dies kann nur in Absprache mit den Prozessverantwortlichen erfolgen.

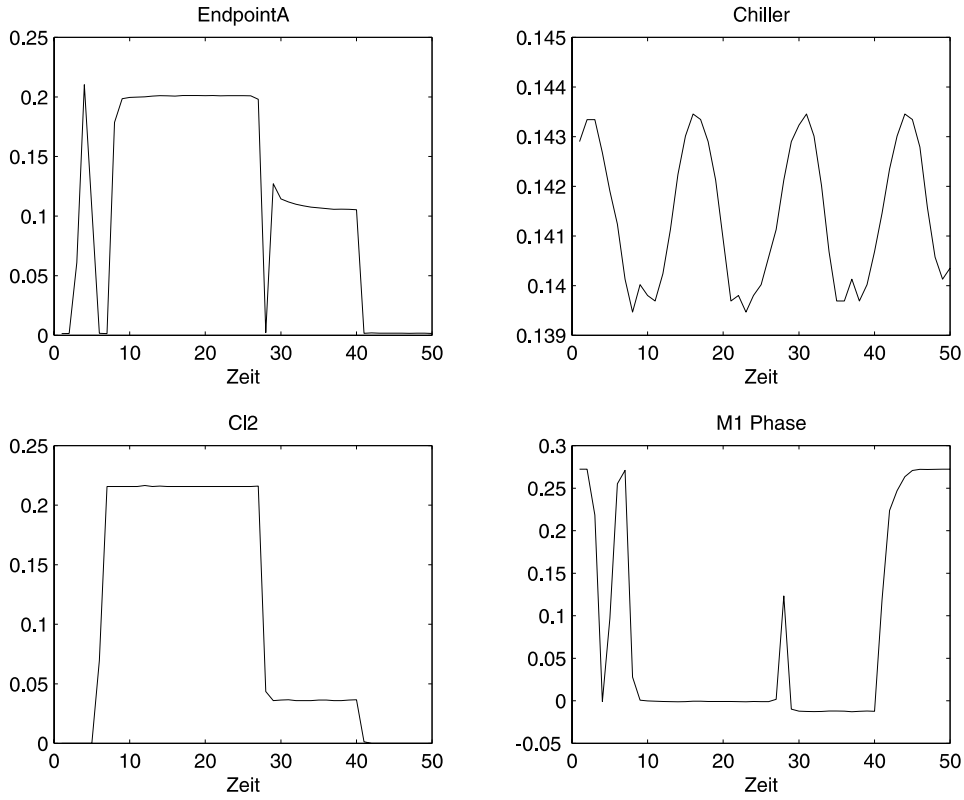

Abbildung 3.1: Unterschiedliche Zeitreihen einer Trockenätzanlage. Gezeigt sind EndpointA (Wellenlängenintensität), Chiller-Bath-Temperatur, Cl2-Konzentration und M1-Phase-Error.

Pro Anlage werden bis zu 70 Signalverläufe mit einer Samplingrate von 1 Hz aufgezeichnet. Die Herausforderung besteht nun darin, diese Datenflut in vernünftiger Weise auszuwerten. Bei einer Vielzahl von Signalverläufen ist dabei nicht bekannt, ob es sich hierbei um gute oder schlechte Prozessverläufe handelt. Die Überlegungen, trotzdem eine sinnvolle Auswertung zu machen, sind, dass man aufgrund der hohen Ausbeute von über 90% davon ausgehen kann, dass die Prozessierungen normalerweise gut verlaufen und Fehlprozessierungen „Ausreißer“ darstellen. Bereiche im hochdimensionalen Phasenraum mit

einer hohen Datendichte müssten also Bereiche guter, Ausreißer, schlechte oder zumindest abnormale Prozessierungen indizieren.

Die einzelnen Messkurven sind zeitabhängige Kurven, wobei die Prozessierungsdauer zwischen 280 und 500 Sekunden liegt. Zur späteren Vergleichbarkeit werden diese Zeitreihen einheitlich skaliert. Für die folgenden Untersuchungen wurden die Zeitreihen durch eine lineare Regression auf 50 Datenpunkte genähert. Anschließend werden die einzelnen Messkurven auf Eins normiert.

Bei einigen Verfahren, wie PCA und ICA ist es nötig, den Mittelwert der Messkurven von den einzelnen Signalverläufen zu subtrahieren. Der Erwartungswert der zu untersuchenden Datensätze muss bei diesen Verfahren gleich Null sein, was bereits im Theorieteil erläutert wurde.

3.1.2 Dimensionsreduzierung mittels PCA

In den theoretischen Grundlagen wurden verschiedene Methoden zur PCA dargestellt. Im Folgenden werden sowohl die klassische Methode, d.h. Diagonalisieren der Kovarianzmatrix, als auch die neuronalen Ansätze, d.h. GHA- und APEX-Algorithmen verwendet und miteinander verglichen. Die PCA wird sowohl als Vorverarbeitungsschritt eingesetzt, d.h. nach der PCA wird als weiteres neuronales Netz ein ICA-Netzwerk oder ein Kohonennetzwerk, als auch als eigenständige Methode zur Kennzahlenextraktion verwendet.

In den folgenden Untersuchungen werden die sogenannten *EndpointA*-Zeitreihen verwendet. Es stehen 1695 verschiedene Messkurven, wie in Abbildung 3.1 ersichtlich, zur Verfügung. Diese Messdaten bestehen jeweils aus 50 Datenpunkten und bilden dadurch eine Punktmenge in einem 50-dimensionalen Phasenraum. Die PCA wird zuerst klassisch durch die Diagonalisierung der Kovarianzmatrix von 1695 Punkten im 50-dimensionalen Raum, berechnet. In Abbildung 3.2 sind die ersten fünf Hauptachsen dargestellt.

Die erste Hauptachse detektiert Bereiche größter Varianz in den Rohdaten. Auffällig sind die Flanken für den sogenannten *Breakthru* zu Beginn der Ätzung, die Varianz bzgl. des sogenannten *Mainetch,* und die Bereiche des sogenannten *Overetch* am Ende der Prozessierung. Die weiteren Hauptachsen geben zusätzliche Informationen über Informations- bzw. Varianzverteilungen in den Rohdaten.
Für eine erste qualitative Vergleichbarkeit wurden die ersten fünf Hauptachsen mit dem GHA Algorithmus bestimmt und in Abbildung 3.3 dargestellt.
Die ersten beiden Hauptachsen sind, bis auf marginale Unterschiede, identisch, die dritte und vierte Hauptachse bis auf den Faktor –1 nahezu identisch, und die fünfte Hauptachse weist sichtbare Unterschiede zu den klassisch berechneten Hauptachsen auf.

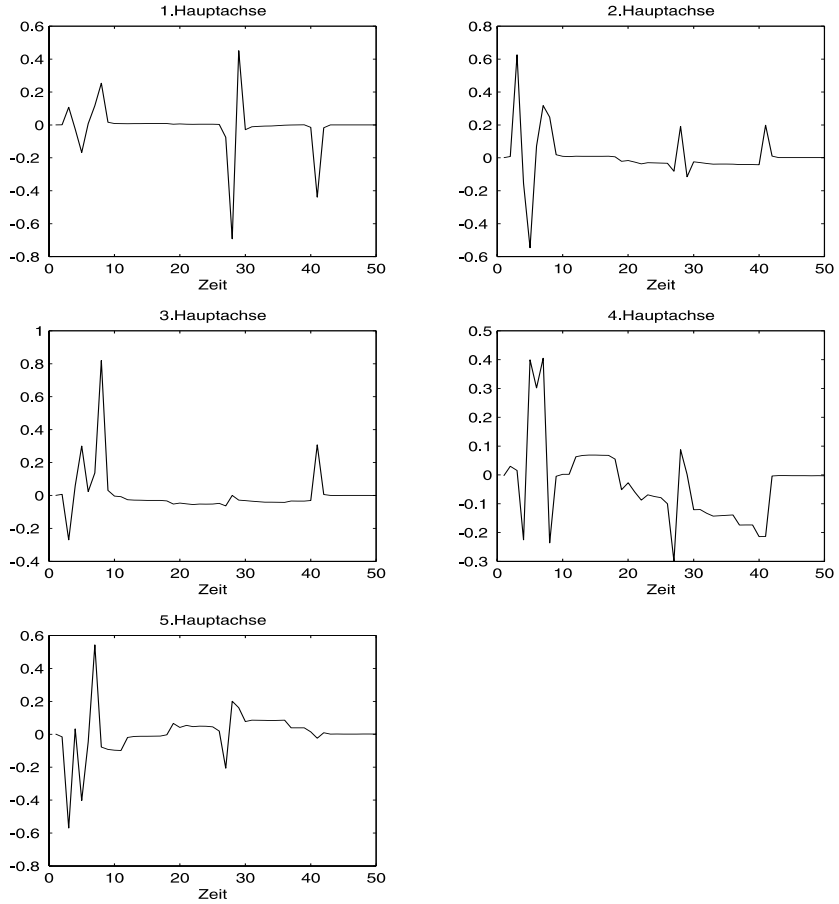

Abbildung 3.2: Die ersten fünf Hauptachsen der EndpointA-Kurven. Die Berechnung erfolgte über die Diagonalisierung der Kovarianzmatrix. Man sieht, dass die erste Hauptachse einige wenige für nahezu alle Messkurven gültige Merkmale detektiert, während die nachfolgenden Hauptachsen auch seltener auftretende Merkmale detektieren.

Die Differenzen lassen sich dadurch erklären, dass beim Diagonalisieren der Kovarianzmatrix die PCA-Hauptachsen exakt berechnet werden. Die Konvergenz der neuronalen Netze hingegen wird mit einer Lernrate gesteuert, die entscheidenden Einfluss auf die Genauigkeit der Hauptachsen bzw. Hauptkomponenten hat.
Beim GHA-Algorithmus bestimmt das neuronale Netz nur fünf Hauptkomponenten. Wobei die Genauigkeit der kleineren von den zuvor berechneten größeren Hauptachsen abhängig ist.

Die Berechnungsfehler größerer Hauptachsen summieren sich dadurch in niedrigeren Hauptachsen auf. Dieses Verhalten ist auch bei der Betrachtung der Eigenwerte der zugehörigen Hauptachsen und damit der Varianzen der Hauptkomponenten gegeben. Die Varianzen der einzelnen Hauptkomponenten wurden durch die Diagonalisierung der Kovarianzmatrix, sowie mit dem GHA- und APEX-Algorithmus berechnet, und sind in Tabelle 3.1 dargestellt.

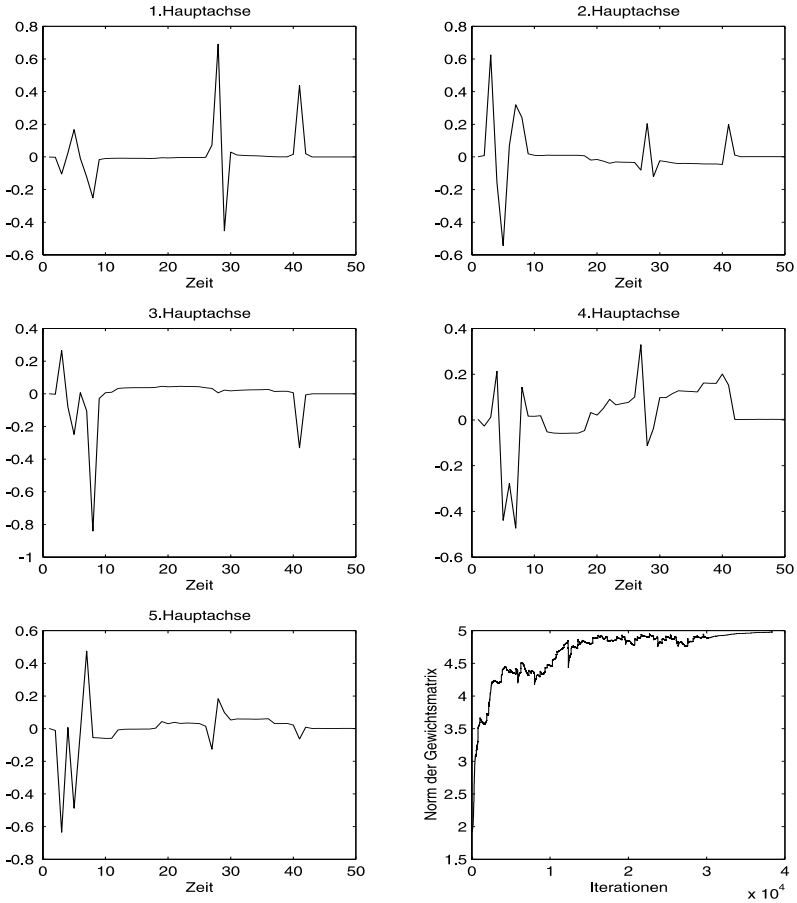

Abbildung 3.3: Die ersten fünf Hauptachsen der EndpointA-Kurve, die mit GHA berechnet wurden. Die ersten Hauptachsen sind sehr ähnlich zu den klassisch berechneten in Abbildung 3.2. Bei der fünften Hauptachse sind allerdings signifikante Unterschiede erkennbar. Zusätzlich ist rechts unten die Norm der Gewichtsmatrix abgebildet, die ein Maß für die Konvergenz des neuronalen Netzes darstellt.

Tabelle 3.1: Mit verschiedenen Algorithmen berechnete Varianzen der Hauptkomponenten

	Diagonalisierung	GHA	APEX
1. Principal Component	$4.24 \cdot 10^{-3}$	$4.24 \cdot 10^{-3}$	$4.23 \cdot 10^{-3}$
2. Principal Component	$2.16 \cdot 10^{-3}$	$2.17 \cdot 10^{-3}$	$2.11 \cdot 10^{-3}$
3. Principal Component	$8.97 \cdot 10^{-4}$	$8.94 \cdot 10^{-4}$	$8.06 \cdot 10^{-4}$
4. Principal Component	$7.76 \cdot 10^{-4}$	$7.55 \cdot 10^{-4}$	$8.77 \cdot 10^{-4}$
5. Principal Component	$3.72 \cdot 10^{-4}$	$3.60 \cdot 10^{-4}$	$3.76 \cdot 10^{-4}$

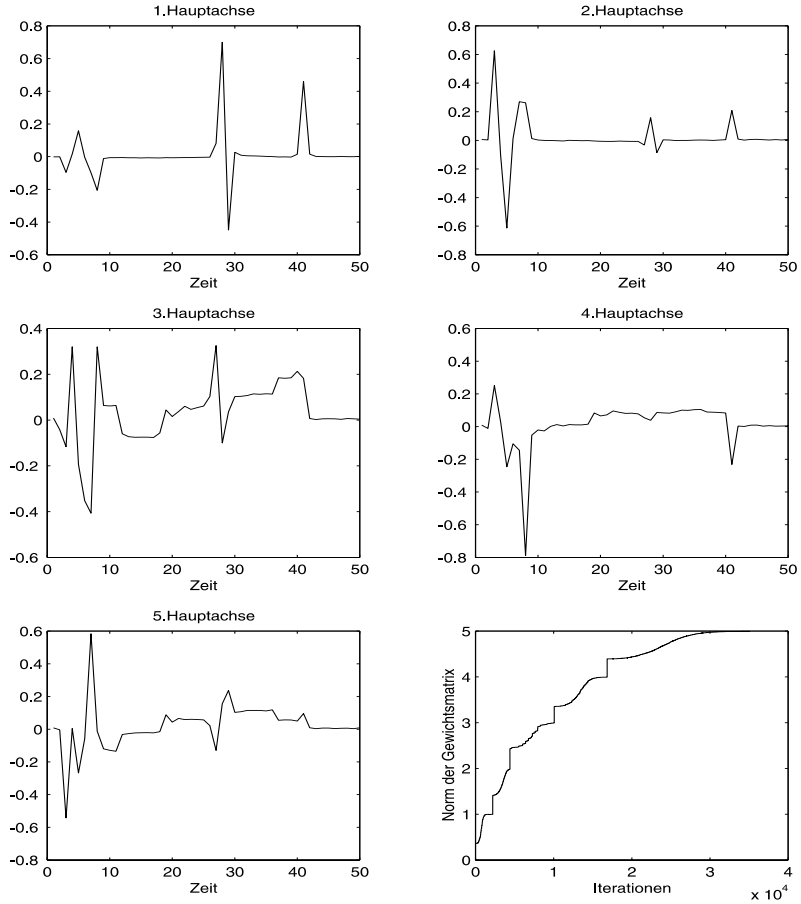

Abbildung. 3.4: Bestimmung der ersten fünf Hauptachsen der EndpointA-Kurven durch den APEX-Algorithmus. Die erste Hauptachse stimmt ziemlich gut mit der ersten klassisch berechneten Hauptachse überein. Bereits bei der zweiten Hauptachse treten kleine Abweichungen auf. Die dritte und vierte Hauptachse sind noch nicht konvergiert, wie man im Vergleich zu den klassisch berechneten Hauptachsen sieht. Rechts unten ist die zeitliche Entwicklung der Norm der Gewichtsmatrix abgebildet, die auch bei diesem Algorithmus ein Maß für die Konvergenz darstellt.

Die Varianzen der ersten beiden Hauptkomponenten sind bei der klassischen Methode und dem GHA-Algorithmus nahezu identisch. Erst bei höheren Eigenvektoren ergeben sich Abweichungen.

Als weitere Methode wurde der APEX-Algorithmus untersucht. Die Abweichungen sind etwas größer als beim GHA-Algorithmus. In Abbildung 3.4 sind die beim APEX-Verfahren ermittelten Hauptachsen zu sehen. Die Entwicklung der Norm der Gewichtsmatrix ist ebenfalls abgebildet. Die Plateaus entstehen dadurch, dass die Hauptachsen erst konvergieren müssen, bevor ein neues Ausgabeneuron hinzugefügt und somit eine weitere Hauptachse berechnet wird.

Da die Varianzen beim APEX-Algorithmus nicht in fallender Reihenfolge angeordnet sind, sieht man außerdem, dass die Berechnung nicht korrekt beendet wurde (siehe dritte und vierte

Varianz in Tabelle 3.1). Dies macht sich auch in den Hauptachsen bemerkbar, da hier die dritte und vierte Hauptachse, im Vergleich mit den anderen PCA-Verfahren, vertauscht ist. Dies bedeutet, dass der APEX-Algorithmus in diesem Fall noch nicht korrekt konvergiert ist und nur die ersten beiden Hauptachsen richtig ermittelt wurden.

Der Vergleich der Rechenzeiten der verschiedenen Algorithmen zeigt, dass die Diagonalisierung der Kovarianzmatrix die schnellste Methode darstellt. Dies ist v.a. auch deshalb interessant, da dieser Algorithmus alle Hauptkomponenten bestimmt, die GHA und APEX-Methode (zumindest in diesem Beispiel) allerdings nur die ersten fünf.

Tabelle 3.2: Rechenzeit für PCA-Methoden

	Diagonalisierung	GHA	APEX
Rechenzeit	3 Sekunden	20 Minuten	5 Minuten

Dieses Verhältnis ändert sich, wenn die Eingabedimensionalität sehr viel größer als die Ausgabedimensionalität ist. In diesem Beispiel, falls die Dimensionalität der Zeitreihen viel größer ist, als die Anzahl zu bestimmender Eigenvektoren. In solchen Fällen kann es günstig sein, die PCA durch neuronale Netze durchführen zu lassen.

Neben der Detektion systeminhärenter Information durch die Betrachtung der Varianzen liegt der Sinn der PCA hauptsächlich in der Dimensionsreduzierung hochdimensionaler Datensätze. Dazu wird im Folgenden eine Fehlerabschätzung für die Reduzierung des 50-dimensionalen *EndpointA*-Datensatzes auf fünf Dimensionen gemacht. Es werden alle 1695 *EndpointA*-Kurven auf die ersten fünf Hauptachsen projiziert und anschließend zurücktransformiert. Die Rekonstruktion lässt sich schreiben als

$$\mathbf{X}_{rec} = \mathbf{W}^T \mathbf{W} \mathbf{X}, \qquad \text{mit} \quad \mathbf{W} \in \Re^{5x50} \tag{3.1}$$

Da $\boldsymbol{W}$ wegen der Eigenvektoren in den Spalten orthogonal ist, lässt sich statt der Pseudoinversen die transponierte Matrix von $\boldsymbol{W}$ schreiben. In Abbildung 3.5 sind zwei Beispiele der Rekonstruktion dargestellt. Die Rekonstruktion ist bereits mit den ersten fünf Hauptkomponenten sehr gut, was bedeutet, dass nahezu die gesamte Information des 50-dimensionalen Datensatzes in diesen Komponenten gespeichert ist. Als quantitatives Maß für die Rekonstruktion wird der mittlere quadratische Rekonstruktionsfehler verwendet, also

$$err = \frac{1}{50} \sqrt{\sum_{i=1}^{50} (x_{rec}(i) - x(i))^2} \tag{3.2}$$

In Abbildung 3.6 ist der Rekonstruktionsfehler für alle 1695 Zeitreihen dargestellt. Der Rekonstruktionsfehler ist unter Verwendung von fünf Hauptachsen bereits sehr gering. Der durchschnittliche Fehler liegt bei ca. $6{,}3 \cdot 10^{-4}$. Für die anderen Zeitreihen wie Cl2, M1 Phase

etc. erhält man ähnliche Werte. Auch hier ist die meiste Information in den ersten fünf Hauptkomponenten enthalten.

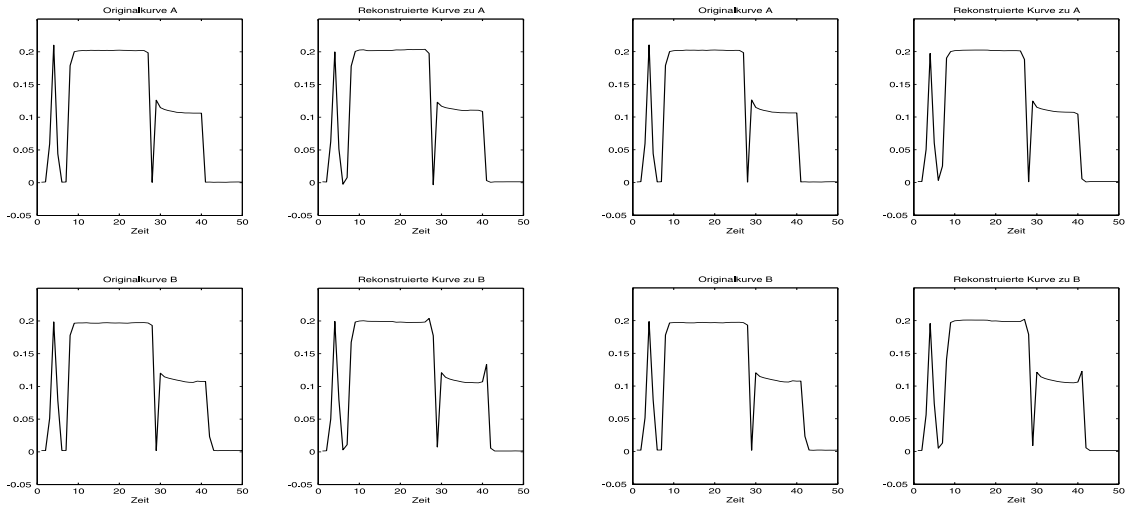

Abbildung 3.5: Rekonstruktion der EndpointA-Signalverläufe unter Verwendung von (links) fünf und (rechts) zwei Hauptkomponenten.

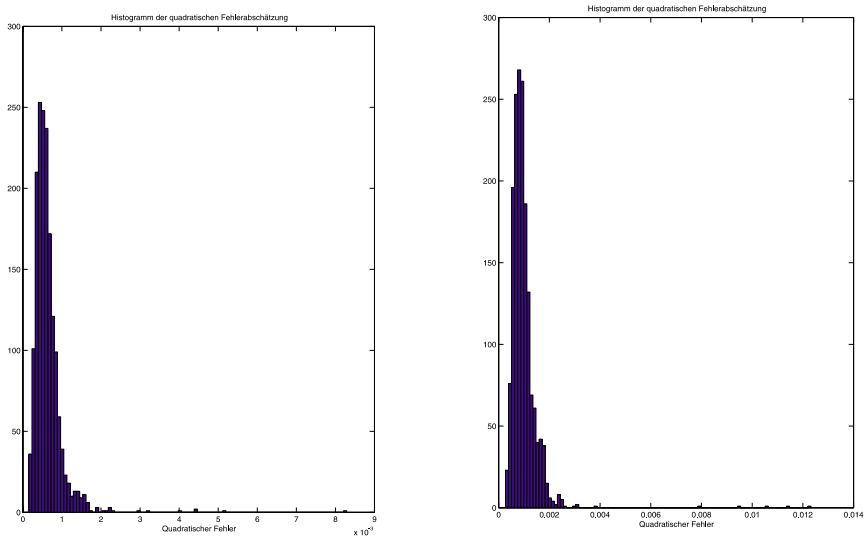

Abbildung 3.6: Quadratischer Rekonstruktionsfehler bei (links) Verwendung von 5 Hauptkomponenten und (rechts) von zwei Hauptkomponenten. Die meisten Rekonstruktionen liefern einen kleinen quadratischen Fehler. Bei einigen ist allerdings der Rekonstruktionsfehler bis zu achtmal größer als der mittlere quadratische Rekonstruktionsfehler.

In Abbildung 3.5 sind Originalkurven und ihre rekonstruierten Kurven unter Verwendung von zwei Hauptkomponenten zu sehen. Einen quantitativen Überblick gibt auch hier wieder die Darstellung des mittleren quadratischen Rekonstruktionsfehlers in Abbildung 3.6. Der durchschnittliche Fehler liegt bei $1{,}0 \cdot 10^{-3}$ im Vergleich zu $6{,}3 \cdot 10^{-4}$ bei fünf Hauptkomponenten.

Die Dimensionsreduzierung auf zwei Hauptachsen ist deshalb von Bedeutung, da zweidimensionale Datensätze leichter visualisiert und analysiert werden können als höherdimensionale Räume. Aus diesem Grund wird im nächsten Abschnitt, bei der Anwendung von Kohonennetzen die PCA dazu eingesetzt, die Dimensionalität der Datensätze auf Zwei zu reduzieren. Die Berechnung erfolgt dabei auf die klassische Weise, durch die Diagonalisierung der Kovarianzmatrix.

3.1.3 Erstellung von Kohonenkarten

Eine Möglichkeit, auffällige Wafer zu detektieren, ist, Kohonennetze zu verwenden. Um diese Netze grafisch sichtbar zu machen und einfach auswerten zu können, wird die Eingabedimensionalität der Netze mittels PCA auf Zwei reduziert. Das verwendete Kohonennetz ist in Abbildung 2.7 zu sehen. Die Eingabeschicht besteht aus zwei Eingabeneuronen, die vollständig mit der Ausgabeschicht vernetzt sind. Die Ausgabeschicht besteht aus 20 x 20 Neuronen, die man sich als Knoten in einem Gitter vorstellen kann.
Von den insgesamt 1695 Datensätzen werden 2/3 zum Training und das restliche Drittel zum Testen des konvergierten Netzwerkes verwendet. Ein typisches Beispiel ist in Abbildung 3.7 links zu sehen.

Im rechten Teilbild von Abbildung 3.7 ist die Dichteverteilung im Eingaberaum zu sehen. Es handelt sich hierbei um ca. 1100 Datenpunkte bzw. *EndpointA*-Messkurven, die durch PCA auf zwei Dimensionen reduziert wurden. Links ist das konvergierte Kohonennetz, bestehend aus 20 x 20 Neuronen zu sehen. Die 400 Ausgabeneuronen nähern die Dichteverteilung im Eingansraum sehr gut an. Auffällig ist, dass sich ein großer Cluster bildet, der an den Rändern eine geringere Dichte und in Richtung der positiven Ordinate einige deutliche Ausreißer aufweist. Diese Darstellung zeigt die große Übereinstimmung mit der Vermutung, dass sich die Prozessdaten in mehr oder weniger komplexen Strukturen im Phasenraum häufen und sich wenige Wafer als Ausreißer präsentieren. Diese abnormalen Signalverläufe weisen auf abnormal prozessierte Wafer hin. Ob sich diese Abnormität tatsächlich als qualitätsrelevant erweist, kann nur durch eine Korrelation mit den Funktionstestdaten ermittelt werden. Diese Datenzuordnung steht, wie in der Einleitung bereits beschrieben, zur Zeit leider nicht zur Verfügung und kann deshalb nicht durchgeführt werden. Diese Korrelation ist allerdings nicht unbedingt notwendig, denn jede Abnormität, ob nun qualitätsrelevant oder nicht, sollte untersucht werden, um die Anlage so stabil zu halten, damit die den Prozess charakterisierenden Prozessvariablen im gleichen Bereich des Phasenraums sind und somit alle Wafer als normal und somit „gut“ prozessiert sind.

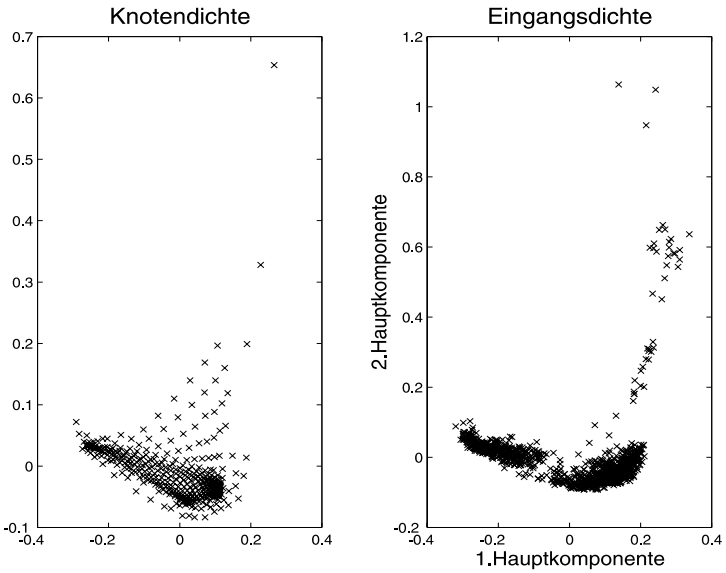

Abbildung 3.7: Rechts ist die Dichteverteilung im Eingaberaum und links die Knotendichte des Kohonennetzwerkes nach Beendigung des Trainingsvorganges dargestellt. Verwendet wurden die ersten beiden Hauptkomponenten des EndpointA-Datensatzes.

Im wesentlichen zeigen alle 10 untersuchten Prozessvariablen ähnliche Kohonennetze. Einzige Ausnahme bildet der Parameter Chiller-Bath-Temperatur, der in Abbildung 3.8 zu sehen ist. Hier sieht es so aus, als ob es eine ziemlich homogene Dichteverteilung im Eingaberaum gibt. Es gibt zwar eine Art Cluster, aber keine exakt definierbaren Randgebiete, in denen die auffälligen Signalverläufe liegen. Anders als bei den *EndpointA*-Signalverläufen gibt es keine auffälligen Ausreißer.

Bei der genaueren Untersuchung der Kohonenkarte fällt auf, dass in der Mitte dieses ellipsoiden Clusters viele sogenannte tote Neuronen liegen. Tote Neuronen sind dabei Neuronen, die beim Lernvorgang nie aktiviert wurden. Um diese Neuronen zu verhindern (sie verfälschen eigentlich die entsprechende Kohonenkarte), können wachsende neuronale Netze verwendet werden. Im Kapitel 3.3 werden u.a. auch diese Netze zur Klassifizierung hochdimensionaler Phasenräume untersucht.

Die Definition des „Ausreißers“ ist i.A. keine einfache Aufgabe und kann in unserem Fall entweder empirisch oder heuristisch versucht werden. Im Rahmen dieses Teilprojektes wurden die drei Methoden, Nächste-Nachbar, Rasterung der Kohonenkarte und empirische univariate Grenzwertprüfung, verwendet.

Bei der Nächste-Nachbar-Methode wird die mittlere euklidische Distanz der einzelnen Neuronen berechnet und als Cluster-Kriterium bzw. zur Ausreißerdefinition verwendet. Für die Kohonenkarte von *EndpointA*-Kurven in Abbildung 3.9 wurde die Nächste-Nachbar-Methode angewendet und zeigt für eine Grenze von 0,02 die Neuronen, die zum Cluster gehören (Kreuz) bzw. die Ausreißer (Kreis). Problematisch ist hier natürlich die Bestimmung des Grenzwertes, der willkürlich auf 0,02 gesetzt wurde und in der Praxis nur in Absprache mit Erfahrungsträgern des Prozesses oder tatsächlich durch Korrelation mit den Prozessergebnissen festgelegt werden kann.

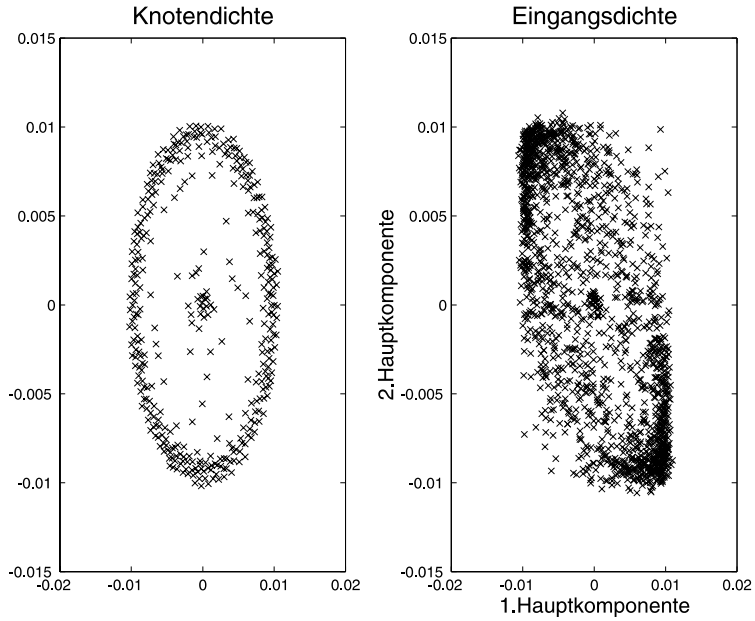

Abbildung 3.8: Rechts ist die Dichteverteilung im Eingaberaum und links die Knotendichte des Kohonennetzes nach Beendigung des Trainingsvorganges zu sehen. Verwendet wurden die ersten beiden Hauptkomponenten des Chiller Bath Temperatur Datensatzes.

Die „Rasterung“ der Kohonenkarte stellt eine weitere Möglichkeit der Ausreißerdetektion dar. Die Gebiete, in denen die Neuronen der Karte liegen, werden im zweidimensionalen quadratisch, bei höheren Dimensionen in entsprechende Segmente, unterteilt. Liegt die Neuronendichte unter einem bestimmten Grenzwert, repräsentiert dieses Segment ein Randgebiet. Liegen die Neuronen in einem Randgebiet, so detektieren sie mit einer Wahrscheinlichkeit von ~1/Neuronendichte auffällige Wafer.

In Abbildung 3.9 ist eine solche Kohonenkarte dargestellt. Die Karte wird in 7x8 Felder eingeteilt, in denen die entsprechenden Neuronendichten ermittelt werden. Liegen weniger als 7 Neuronen in einem solchen Segment, wird dieses Gebiet als Randgebiet deklariert.

In der Tabelle 3.3 sind einige Lose aufgeführt, bei denen mit der Nächste-Nachbar- und mit der Rasterungsmethode auffällige Wafer detektiert wurden. Zur Untersuchung wurden zwei Lose mit je 50 Wafer dem trainierten Netzwerk präsentiert und klassifiziert. Auffällig ist dabei, dass die Wafer, die durch die Rasterungsmethode als auffällig klassifiziert wurden, nahezu alle auch durch die Nächste-Nachbar-Methode klassifiziert wurden. Dies ist auch anhand der Kohonenkarten in Abbildung 3.9 ersichtlich, da dort die Neuronen der Rastermethode, die die Randgebiete abdecken, eine Untermenge der Randgebietneuronen der Nächste-Nachbar-Methode sind.
Einige der auffälligen *EndpointA*-Kurven des Loses 4050 sind in Abbildung 3.10 zu sehen.

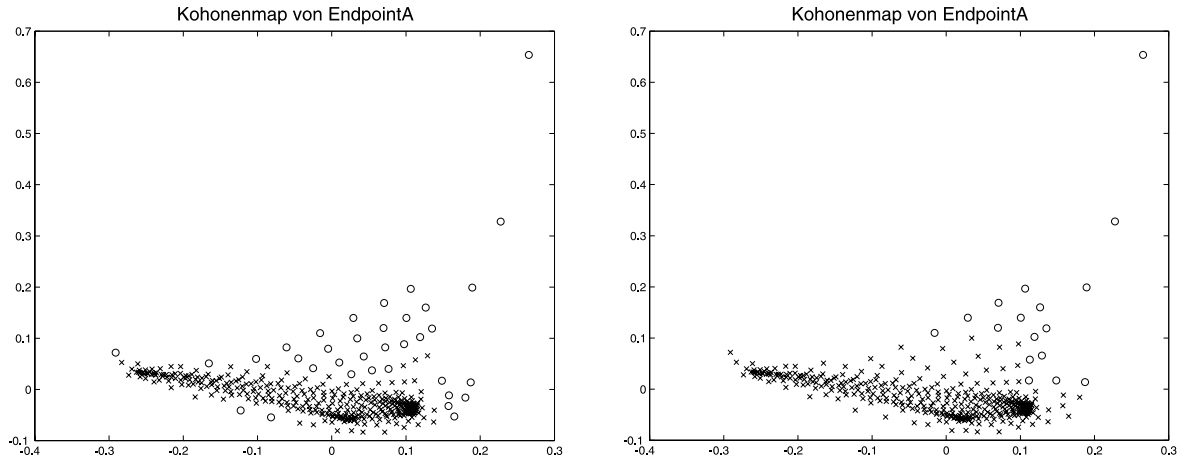

Abbildung 3.9: Ausreißerbestimmung mit Nächster-Nachbar-Methode (links) und Rasterung der Kohonenkarte (rechts). Beide Bilder stellen trainierte Kohonenkarten der ersten beiden Hauptkomponenten des EndpointA-Datensatzes dar. Neuronen als Kreuz dargestellt repräsentieren den Cluster, Neuronen mit Kreis die Randgebiete bzw. die Ausreißer.

Tabelle 3.3: Auffällige Wafer bei Los 4050 und 3587

Methode	Rasterung		Nächste-Nachbar	
Losnummer	3587	4050	3587	4050
Wafer	27	25	23	10
	48	34	27	13
		46	40	14
		49	48	25
				41
				46
				49

Zum Vergleich der auffälligen Kurven mit einem normalen Prozessierungsverlauf ist ein Prototyp der Messkurven dargestellt, d.h. eine Zeitreihe, die im Cluster der unauffälligen Messkurven liegt. Abnormitäten ergeben sich u.a. durch eine Stauchung der Zeitreihe, d.h. eine auffällig kürzere Prozessierungsdauer. Da bei diesem Ätzvorgang die Anlage automatisch das Prozessierungsende durch einen Endpunkterkennungs-Algorithmus erkennen soll, liegt diese kürzere Prozessierungsdauer entweder darin, dass die wegzuätzende Schicht dünner oder die Ätzrate größer als üblich ist. Eine weitere Ursache könnte in der falschen Endpunkterkennung liegen, d.h. die Anlage stoppt die Ätzung, obwohl die abzutragende Schicht noch nicht vollständig entfernt wurde. Im Rahmen dieser Arbeit ist die Ursache für die unterschiedlichen Verläufe allerdings eher nebensächlich. Ausschlaggebend ist hauptsächlich, dass es Bereiche im Phasenraum gibt, in denen die überwiegende Mehrzahl der Prozessierungen clustert und einzelne Ausreißer die Ausnahme darstellen, die nach der Detektierung von Prozessspezialisten separat untersucht werden müssen.

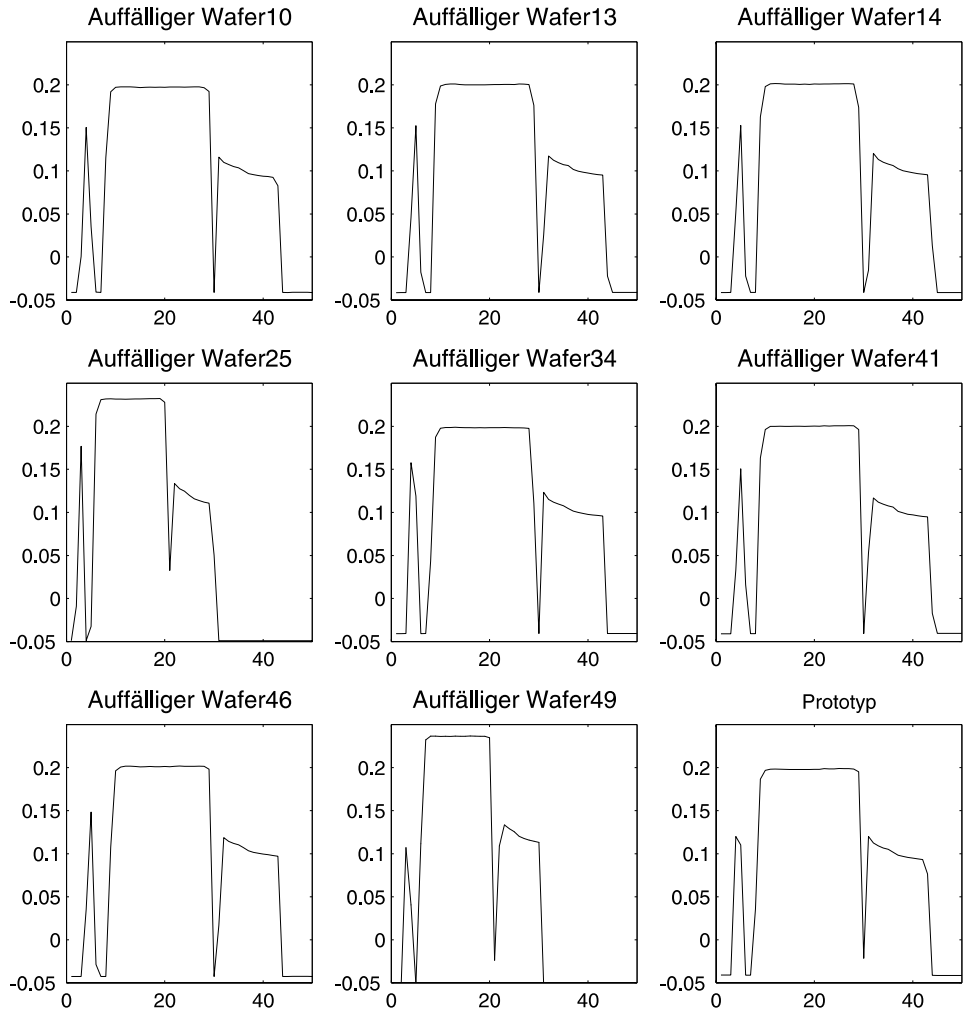

Abbildung 3.10: Die durch die Kohonennetze als auffällig klassifizierten Wafer des Loses 4050, rechts unten ist ein im Cluster liegender, also unauffälliger Wafer zu sehen. Die dargestellten Zeitreihen sind die EndpointA-Prozessparameter.

Um die Abhängigkeit der Kohonenkarte und damit die detektierten Abnormitäten von der Anzahl der Hauptkomponenten zu untersuchen, wurde ein Kohonennetz mit fünf Eingabeneuronen trainiert. Die verwendeten Zeitreihen waren wiederum die *EndpointA*-Datensätze. Zur Visualisierung des konvergierten Netzwerkes werden zweidimensionale Projektionen der Gewichtsmatrix verwendet. In Abbildung 3.11 sind die 10 möglichen Schnitte der insgesamt fünfdimensionalen Matrix gezeigt. Zur Ausreißerdetektion wurde die Nächste-Nachbar-Methode verwendet. Die durch Kreise dargestellten Neuronen repräsentieren die Randgebiete. Dem konvergierten Netzwerk werden Lose mit je 50 Wafer zur Klassifikation präsentiert.

Tabelle 3.4: Auffällige Wafer bei Los 3587

Schnitte	1,2	1,3	1,4	1,5	2,3	2,4	2,5	3,4	3,5	4,5
	40	34	10	4	48	27	27	3	6	4
	48	39	44	6		48	48	6	15	6
		40	48	13				25	23	23
		43		21				27	25	25
		44		23				29	27	27
		46		24				34	29	34
		47		27				38	34	40
		48		28				39	39	43
				40				43	43	47
				47				46	46	48
				48				47	47	
								48	48	

Tabelle 3.5: Auffällige Wafer bei Los 4050

Schnitte	1,2	1,3	1,4	1,5	2,3	2,4	2,5	3,4	3,5	4,5
	1	1	4	1	14	4	13	25		1
	10	2	13	2	19	13	14	49		25
	25	3	14	3	25	14	25			26
	32	5	18	6	35	18	41			32
	33	6	19	7	36	19	49			33
	49	7	20	9	49	20				49
		9	21	10		25				
		10	25	11		35				
		11	29	12		36				
		12	35	13		38				
		17	36	14		41				
		22	41	16		46				
		25	46	17		49				
		26	49	19						
		27		21						
		31		22						
		32		23						
		33		25						
		37		26						
		39		27						
		40		28						
		44		30						
		45		31						
		48		32						
		49		33						
				37						
				40						
				41						
				44						
				48						
				49						

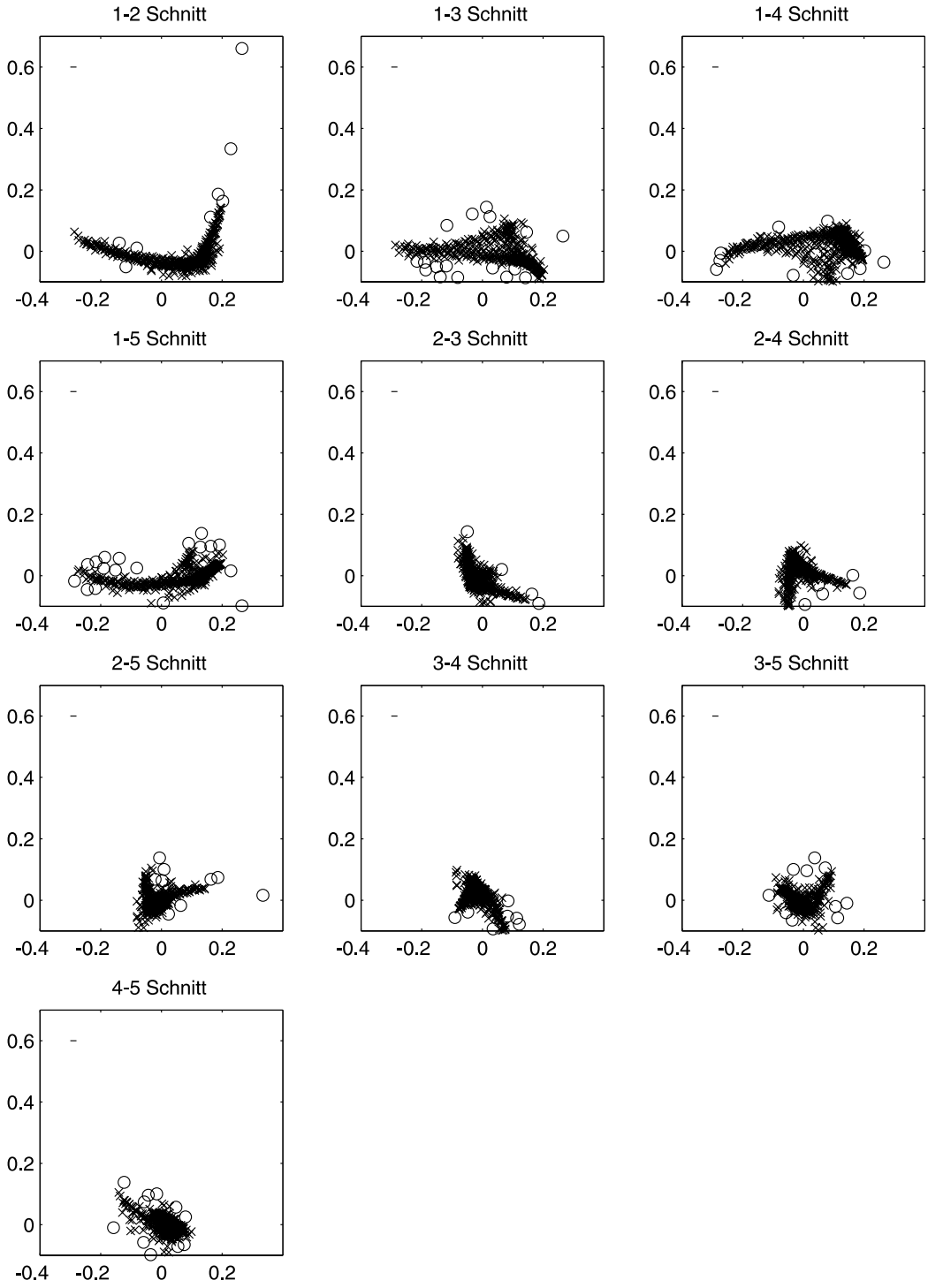

Abbildung 3.11: Kohonennetz mit fünf Eingabeneuronen, das mit dem EndpointA-Datensatz trainiert wurde. Dargestellt sind die 10 möglichen zweidimensionalen Schnitte des Kohonennetzes. Als Kreise dargestellte Neuronen detektieren auffällige Messkurven.

In Tabelle 3.4 sind die als auffällig deklarierten Wafer aufgelistet. Beim Testlos 3587 zeigen sich bei der Berücksichtigung von fünf Hauptkomponenten mehr Auffälligkeiten als bei der Analyse und Klassifikation mit nur zwei Eingabeneuronen. Dies liegt v.a. daran, dass, wie in Abbildung 3.11 ersichtlich, Bereiche im Phasenraum existieren, die keine eindeutig separierten Ausreißer aufweisen und somit auffällige und normale Zeitreihen schlecht separieren. Diese Schnitte eignen sich deshalb weniger zur Detektion von auffälligen Wafern. Noch besser erkennt man dies beim Los 4050, bei dem manche Schnitte nahezu alle Wafer als auffällig detektieren, wie in der Tabelle 3.5 für Schnitt 1,5 und 1,3 gemessen wurde.

Zur Analyse hochdimensionaler Räume gibt es eine Vielzahl weiterer Methoden. So wäre die zur Detektierung interessanter Projektionen vorgeschlagene Exploratory Projection Pursuit (EPP) [Huber85] geeignet, passende, d.h. interessante bzw. Schnitte mit hohem Informationsgehalt im hochdimensionalen Phasenraum zu erkennen. Als interessante Projektionen werden dabei i.d.R. Projektionen genannt, die keiner Gaußverteilung entsprechen. Des weiteren könnten Schnitte mit hoher Kurtosis selektiert werden, was Projektionen entspricht, die einen kleinen lokalisierten Cluster enthalten und weit ausgedehnte Randgebiete geringer Dichte. Im Rahmen dieses Kapitels werden diese Methoden nicht weiter untersucht, da bereits mit niedrigdimensionalen Räumen relevante Auffälligkeiten erkannt werden. Im Kapitel 3.3 werden weitere Verfahren zur Mustererkennung und Klassifikation hochdimensionaler Räume untersucht. Darüber hinaus ist der wirtschaftliche Nutzen bereits enorm, wenn nur die groben Ausreißer unmittelbar nach der Prozessierung detektiert und Maßnahmen zu deren Beseitigung eingeleitet werden. Die Anzahl von gemeldeten Abnormitäten ist auch als kritische Größe für die Umsetzung und Akzeptanz in der Praxis zu bewerten.

3.1.4 Statistische Unabhängigkeit durch ICA

Um statistische Abhängigkeiten höherer Ordnung in den Datensätzen zu separieren, wurde mit den Zeitreihen eine Independent Component Analysis durchgeführt. Im wesentlichen wurden zwei verschiedene Architekturen verwendet, um eine Vergleichbarkeit zu erhalten. Dieser Ansatz der ICA wurde erstmals von Bartlett [Bartlett98] beschrieben und wird derzeit in zahlreichen Forschungsgruppen untersucht.

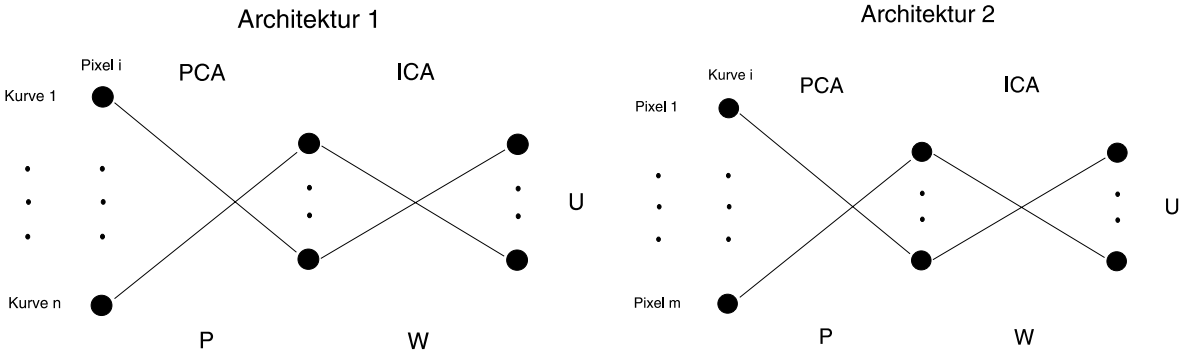

Abbildung 3.12: Darstellung der verwendeten ICA-Architekturen. Architektur 1 liefert statistisch unabhängige Basisbilder, Architektur 2 den Faktorencode. ***P*** *stellt die Gewichtsmatrix des PCA-Netzes dar.* ***W*** *die Gewichtsmatrix des ICA-Netzes und* ***U*** *ist die Ausgabe der neuronalen Netzwerke. Bei Architektur 1 besteht der Eingaberaum aus gleichen Dimensionsvektoren der unterschiedlichen Zeitreihen, in Architektur 2 wird eine gesamte Zeitreihe als Eingabe verwendet.*

Die schematische Darstellung der verwendeten Architekturen ist in Abbildung 3.12 gezeigt. Nach der Eingabeschicht folgt eine PCA-Schicht und anschließend das ICA-Netzwerk. Die Verwendung der PCA wird von verschiedenen Autoren vorgeschlagen. Im Rahmen dieser Arbeit werden die zu untersuchenden Zeitreihen sowohl mit PCA als auch ohne diesen Vorverarbeitungsschritt untersucht. Der Eingaberaum wird als ***X***, die Gewichtsmatrix der PCA als ***P***, die ICA-Entmischungsmatrix ***W*** und die Ausgabe mit ***U*** bezeichnet.

Die zu untersuchende Datenmatrix ***X*** besteht im Folgenden aus 50 Dimensionen (Spalten) und 1695 gemessenen unterschiedlichen Zeitreihen (Zeilen).

Für die Architektur 1 gilt für die Ausgabe **U**:

$$\boldsymbol{U}=\boldsymbol{WPX} \tag{3.3}$$

wobei ***X***: Eingabematrix
P: PCA-Gewichtsmatrix
W: ICA-Entmischungsmatrix
U: Netzwerk-Ausgabe

Für die Rekonstruktion der Eingabevektoren aus der Netzwerk-Ausgabe U gilt:

$$\boldsymbol{X}_{rec}=pinv(\boldsymbol{WP})\boldsymbol{U} \tag{3.4}$$

*pinv(**WP**)* steht für die Pseudoinverse von ***WP***. Falls ***WP*** quadratisch ist, entspricht die Pseudoinverse der Inversion der Matrix. In den Zeilen von ***U*** stehen statistisch unabhängige Basismesskurven, die multipliziert mit den Koeffizienten in *pinv(**WP**)* die Originalkurve rekonstruieren. Die Koeffizienten sind dabei nicht statistisch unabhängig. Diese Architektur findet die statistisch unabhängigen Basismesskurven der Datensätze.

Bei Architektur 2 wird als Eingabe die transponierte Eingabematrix $\boldsymbol{X}^T$ verwendet. Es gilt analog zu (3.3):

$$\boldsymbol{U}=\boldsymbol{WPX}^T \tag{3.5}$$

$$\boldsymbol{X}^T_{rec}=pinv(\boldsymbol{WP})\boldsymbol{U} \tag{3.6}$$

Die Basisbilder der Datensätze sind in den Spalten von *pinv(**WP**)* enthalten. Die Koeffizienten, die multipliziert mit den Basisbildern die Originalkurven ergeben, sind in der Ausgabe ***U*** enthalten. Diese Transformation liefert statistisch unabhängige Koeffizienten und nicht statistisch unabhängige Basisbilder. In der Literatur wird dieser Vorgang als Faktorencode bezeichnet.

Statistisch unabhängige Basismesskurven

Die statistische Unabhängigkeit der Basismesskurven wird durch die Verwendung der Netzwerkarchitektur 1 erreicht. Zum Netzwerktraining wurden verschiedene Lernregeln verwendet. Lernregeln mit fester Skewness und Kurtosis, bei denen die Nichtlinearitäten variiert wurden und Lernregeln mit variabler Skewness und Kurtosis. Dadurch wird die Adaption beider Momente beim Lernvorgang erreicht. Bei den Lernregeln mit adaptierbarer Skewness und Kurtosis muss zwischen den Entwicklungen von Gram-Charlier und Edgeworth unterschieden werden, die zu leicht unterschiedlichen *Score-Functions* führen und im theoretischen Teil behandelt wurden.

Tabelle 3.6: Verwendete Lernregeln bei ICA

Feste Skewness und Kurtosis	Skewness und Kurtosis adaptierbar
y^3 tanh $\frac{3}{4}y^{11}+\frac{15}{4}y^9+\frac{14}{3}y^7-\frac{29}{4}y^5+\frac{29}{4}y^3$	Gram-Charlier Edgeworth

Die Untersuchungen für die Architektur mit den unterschiedlichen Lernregeln zeigte, dass sich die Ergebnisse für die einzelnen Lernregeln nur marginal unterscheiden. Im Folgenden wird deshalb nur die Lernregel mit der Nichtlinearität

$$\phi(y)=\frac{3}{4}y^{11}+\frac{15}{4}y^9+\frac{14}{3}y^7-\frac{29}{4}y^5+\frac{29}{4}y^3$$

verwendet.

Zum Training wurde wiederum die *EndpointA*-Zeitreihe verwendet. Der Einfluss der Vorverarbeitung und Dimensionsreduzierung der Messkurven durch PCA ist in Abbildung 3.15 und 3.16 gezeigt.

Ohne Dimensionsreduzierung durch PCA weisen die ICA-Komponenten bzw. Basismesskurven eine deutlich komplexere Struktur auf, d.h. zahlreiche Peaks und Oszillationen in den Kurven deuten daraufhin, dass auch kleine Abweichungen in den Messkurven detektiert und in den Basismesskurven verstärkt wiedergegeben werden.
Der Vergleich mit den PCA-Hauptkomponenten weist große Unterschiede in den einzelnen Komponenten auf und zeigt, dass signifikante Korrelationen höherer Ordnung in den Zeitreihen enthalten sind

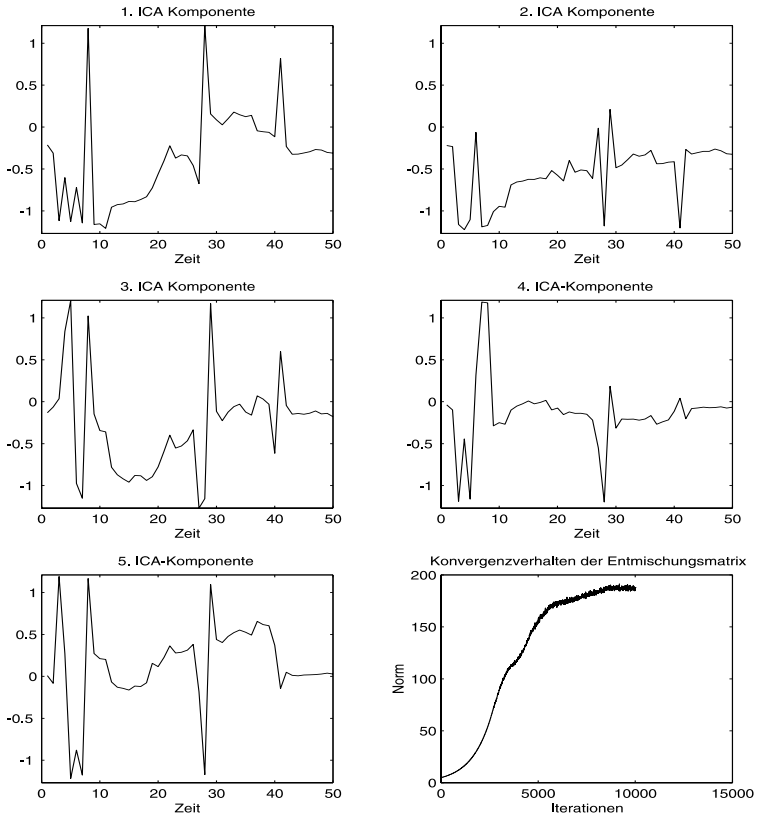

Abbildung 3.15: Fünf berechnete ICA-Komponenten der EndpointA-Datensätze mit der Architektur 1 ohne PCA-Vorverarbeitung. Rechts unten ist die zeitliche Entwicklung der Norm der Entmischungsmatrix als Maß für die Konvergenz zu sehen.

Wird die PCA zur Dimensionsreduzierung als Vorverarbeitung verwendet, ergeben sich durch die ICA-Analyse einfachere Strukturen für die Basismesskurven. Die große Ähnlichkeit der 1. Hauptachse des *EndpointA*-Datensatzes mit der ersten ICA-Komponente ist dabei offensichtlich (siehe Abbildung 3.2 und Abbildung 3.16).
Für diese Untersuchung wurden die ersten beiden Hauptkomponenten als Eingabe für das ICA-Netzwerk verwendet. Da die Varianz der ersten Hauptkomponente um einiges größer ist als die Varianz der zweiten Hauptkomponente, wird von der ICA hauptsächlich die

Information über den Verlauf nur einer Komponente verwendet. Auffällig dabei ist ebenfalls, dass die ICA-Komponenten offensichtlich nicht statistisch unabhängig sind. Dieses visuelle Ergebnis bestätigt sich in der quantitativen Untersuchung, die anschließend durchgeführt wird.

Bartlett et al. [Bartlett98] zeigen, dass durch die Verwendung der Architektur 1 lokale Merkmale aus Datensätzen extrahiert werden. Konkret wurden die Versuche mit Gesichtern gemacht, wobei lokale Informationen wie Lippen, Augen, Nase etc. extrahiert werden konnten.
Für eine anschließende Klassifikation neuer Gesichter, z.B. eines einäugigen Zyklopen, wäre diese Analyse optimal, da dieses Gesicht sofort als anders und somit in unserm Kontext als abnormal erkannt würde.
Dieses Ergebnis konnte in Untersuchungen der nichtlinearen Zeitreihen der Trockenätzanlage im Fall der Vorverarbeitung der Daten mit PCA nicht bestätigt werden. Die Ursachen können dabei vielfältiger Natur sein und liegen wahrscheinlich darin begründet, dass eine zu geringe Zahl von Hauptkomponenten bei der PCA-Analyse verwendet wurden. Diese könnten unter Umständen Korrelationen höherer Ordnung beinhalten und wären für die ICA-Analyse von besonderem Interesse.

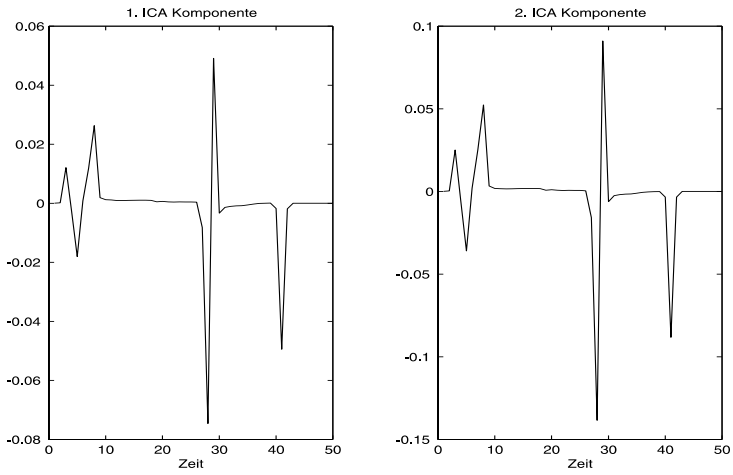

Abbildung 3.16: Zwei berechnete ICA-Komponenten der EndpointA-Datensätze mit Architektur 1 und Dimensionsreduzierung auf zwei Hauptkomponenten durch PCA. Die Komponenten sind, wider erwarten, nicht statistisch unabhängig. Mögliche Ursachen werden im Text erläutert.

Um den Informationsgehalt der einzelnen Basisbilder zu bestimmen, wurde wie bei der PCA eine Rekonstruktion vorgenommen. Das vorrangige Ziel der ICA ist Korrelation höherer Ordnung zu entfernen und nicht die Transformation des Eingaberaumes in Komponenten mit möglichst hoher Varianz. Eine optimale Rekonstruktion der Originaldaten ist deshalb nicht im Fokus der ICA. Die Rekonstruktion wird deshalb nur qualitativ betrachtet.
Bei der Rekonstruktion ergeben sich dabei einige Schwierigkeiten. Die Basiskurven lassen sich nur auf einen Skalierungsfaktor und Permutationen genau bestimmen, wie im theoretischen Teil erläutert wird. Die rekonstruierten Kurven in X_{rec} werden auf Eins normiert

und anschließend muss überprüft werden, welche der rekonstruierten Kurven am besten mit der Originalkurve übereinstimmt.
Dieses Vorgehen wurde für *EndpointA*-Datensätze mit und ohne PCA-Vorverarbeitung durchgeführt. Das Ergebnis ist in Abbildung 3.17 dargestellt.
Im ersten Versuch wurde ein ICA-Netzwerk ohne PCA-Vorverarbeitung und mit fünf Ausgabeneuronen verwendet. Das Rekonstruktionsergebnis ist bereits relativ gut. Dieses Ergebnis lässt sich durch die Vorverarbeitung durch PCA wesentlich verbessern. Als Vorverarbeitung wurden durch die PCA zwei Hauptkomponenten bestimmt, die als Eingabe für das ICA-Netzwerk dienen. Das ICA-Netz enthält sowohl 2 Eingabe als auch 2 Ausgabeneuronen. Da durch die ICA die Dimensionalität nicht verringert wird, bleibt die gesamte, in den PCA-Komponenten enthalte, Information über den Kurvenverlauf erhalten. Dadurch verbessert sich das Rekonstruktionsergebnis, obwohl eine geringere Dimensionalität des Ausgaberaumes vorliegt.

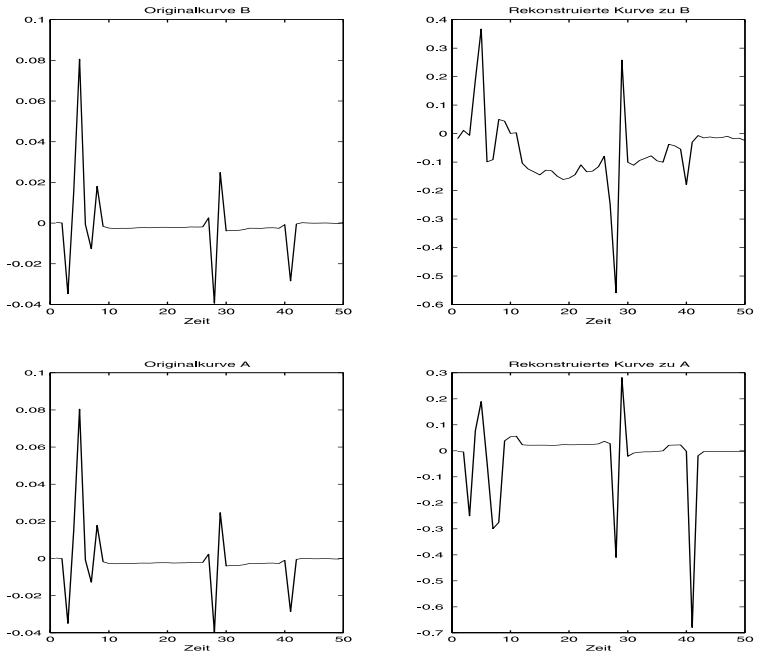

Abbildung 3.17: Qualitative Untersuchung der Rekonstruktionsleistung einer ICA-Analyse. Links oben ist die Originalkurve, rechts daneben die rekonstruierte Kurve dargestellt. Es wurde keine PCA als Vorverarbeitung durchgeführt und fünf ICA-Komponenten berechnet. Links unten original und rekonstruierte Kurve mit PCA-Vorverarbeitung. Durch die PCA wurden 2 Hauptkomponenten bestimmt die als Eingabe für die ICA-Analyse verwendet wurden. Der Ausgaberaum der ICA-Analyse war ebenfalls zweidimensional.

Der eigentliche Nutzen der ICA liegt in der Analyse höherer Korrelationen und der Bestimmung statistisch unabhängiger, d.h. unkorrelierter, Ausgabemuster. Im Folgenden soll deshalb die statistische Unabhängigkeit des Ausgaberaumes untersucht werden.
Wie im theoretischen Teil gezeigt, muss für die statistische Unabhängigkeit zweier Verteilungen **x** und ***y*** folgendes gelten:

$$p(\mathbf{z})=p(\mathbf{x})\cdot p(\mathbf{y}) \tag{3.7}$$

mit

$$\mathbf{z}=(\mathbf{x},\mathbf{y})$$

Dies bedeutet, dass die gemeinsame Wahrscheinlichkeitsdichte *p(**z**)* im Fall statistischer Unabhängigkeit gleich dem Produkt der Einzelwahrscheinlichkeiten *p(**x**)* und *p(**y**)* ist.
Die Aufgabe besteht also darin, die gemeinsame und marginalen Verteilungen zu berechnen. Domany et al. [Domany94] schlägt folgenden Lösungsansatz vor:

Seien $\mathbf{x}=(x_1,\ldots,x_N)$ und $\mathbf{y}=(y_1,\ldots,y_N)$ beliebige Verteilungen
und $\mathbf{z}=(z_1,\ldots,z_N)$
mit $z_i=(x_i,y_i)$.

Man legt ein Raster der Größe ε über den Zustandsraum der Variablen x, y, z und definiert für diese Variablen Wahrscheinlichkeiten in ein Rastergebiet zu fallen mit:

$$p_t(\mathbf{x}) = \frac{1}{N}\sum_{t'=1}^{N}\Theta\left(\varepsilon - \left\|x_t - x_{t'}\right\|\right) \tag{3.8}$$

$$p_t(\mathbf{y}) = \frac{1}{N}\sum_{t'=1}^{N}\Theta\left(\varepsilon - \left\|y_t - y_{t'}\right\|\right) \tag{3.9}$$

$$p_t(\mathbf{z}) = \frac{1}{N}\sum_{t'=1}^{N}\Theta\left(\varepsilon - \left\|z_t - z_{t'}\right\|\right) \tag{3.10}$$

Dabei ist θ die Heaviside-Funktion. Nun kann direkt die *Average Mutual Information* berechnet werden, die als Maß für die statistische Unabhängigkeit der Verteilungen verwendet werden kann.

$$M = \frac{1}{N}\sum_{t=1}^{N} M_t \tag{3.11}$$

mit

$$M_t = \log\frac{p_t(\mathbf{z})}{p_t(\mathbf{x})\,p_t(\mathbf{y})} \tag{3.12}$$

Je statistisch unabhängiger zwei Verteilungen sind, desto kleiner wird die *Average Mutual Information*. Dieses Verfahren lässt sich einfach auf mehr als zwei Verteilungen erweitern.

Dieses Verfahren wurde mit dem BSS Problem, das im Theorieteil beschrieben ist, getestet. Es wurden verschiedene Signale wie Rechtecksignal, Sinussignal u.a. künstlich erzeugt und linear gemischt. Anschließend wurden die gemischten Signale mit einem ICA-Netzwerk entmischt und auf ihre statistische Unabhängigkeit untersucht. In Tabelle 3.7 sind die berechneten Werte aufgelistet. Die Mischungen haben eine größere „*Mutual Information*“ und damit eine geringere statistische Unabhängigkeit als die Originalsignale bzw. separierten Signale.

Tabelle 3.7: Mutual Information bei BSS

	M
Originalsignal	0.795
Mischungen	2.072
Separierte Signale	0.844

Als nächstes wurde dieses Verfahren auf die zu untersuchenden Datensätze angewandt. Es wurde wieder der *EndpointA*-Datensatz verwendet. Um die statistische Unabhängigkeit der Ausgabe der ICA-Netze vergleichen zu können, wurde er mit der statistischen Unabhängigkeit der Ausgabe der PCA-Netze verglichen. Es ergaben sich die Messwerte in Tabelle 3.8 für die statistisch unabhängigen Basiskurven mit der Architektur 1. Es wurden verschiedene Nichtlinearitäten verwendet, wie sie im Theorieteil beschrieben wurden.

Tabelle 3.8: Mutual Information bei Architektur 1

	Nichtlinearität	M
ICA	$\frac{3}{4}y^{11}+\frac{15}{4}y^{9}+\frac{14}{3}y^{7}-\frac{29}{4}y^{5}+\frac{29}{4}y^{3}$	1.593
ICA	tanh	1.229
ICA	y^3	1.668
PCA		1.204

Das Ergebnis ist sehr ungewöhnlich. Die Messwerte zeigen, dass die PCA-Analyse statistisch unabhängigere Ausgabewerte liefert, als die ICA-Netzwerke. Bis auf den ICA-Algorithmus mit der Nichtlinearität *tanh,* der eine in etwa gleiche statistische Unabhängigkeit liefert wie die PCA-Analyse, weisen die anderen ICA-Topologien schlechtere Werte auf.
Eine Ursache dieses unerwarteten Ergebnisses liegt wohl in der großen Differenz zwischen der Zahl der Eingabe- und Ausgabeneuronen. In unserem Experiment wurden 1695 Zeitreihen und somit 1695 Eingabeneuronen und nur fünf Ausgabeneuronen verwendet.
Die Schwierigkeit liegt dabei darin, dass wir bei ICA nicht wissen, in wie viele Basiskurven die Messkurve zerfällt und deshalb die Anzahl der benötigten Ausgabeneuronen unklar ist. Darüber hinaus wird in vielen ICA Veröffentlichen darauf hingewiesen, dass die Zahl der Eingabeneuronen gleich der Ausgabeneuronen sein soll.
In unserem Fall würde das einen 1695-dimensionalen Eingaberaum und einen 1695-dimensionalen Ausgaberaum bedeuten. Die Konvergenz dieser Netzwerkgröße würde einen großen (Rechen-)Zeitaufwand bedeuten und wurde deshalb im Rahmen dieser Arbeit nicht durchgeführt.
Eine weitere Ursache für das von der Theorie abweichende Ergebnis, für die Untersuchung von Architektur 1, könnte darin liegen, dass pro Ausgabeneuron nur 50 Werte für die Berechnung der Wahrscheinlichkeitsverteilung zur Verfügung stehen. Evtl. ist die statistische Basis zur signifikanten Bestimmung der Wahrscheinlichkeiten dadurch nicht gegeben.

Faktorencode

Im Folgenden wird Architektur 2 zur Ermittlung statistisch unabhängiger Koeffizienten näher untersucht. Die Basisbilder, aus denen sich die Originaldaten rekonstruieren lassen, sind dabei nicht statistisch unabhängig und befinden sich in den Spalten von *pinv(**WP**)*.

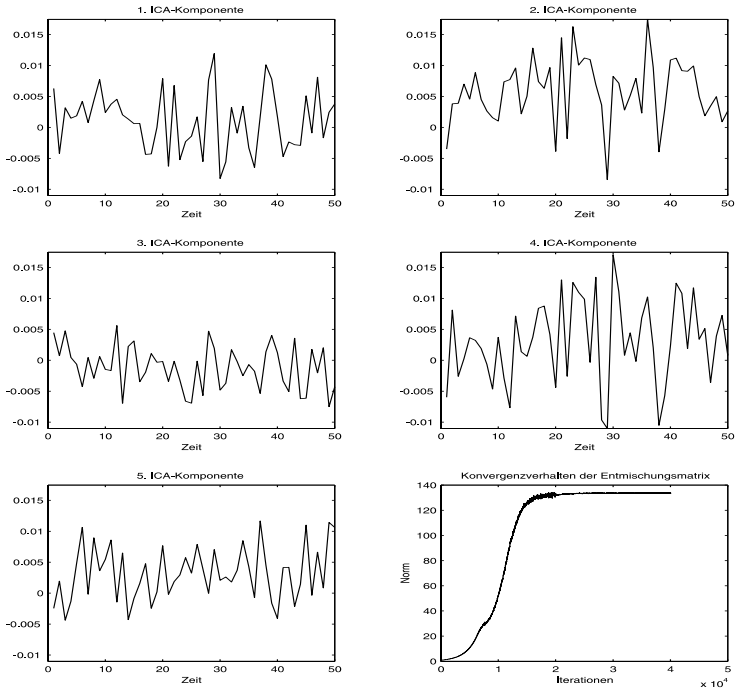

Abbildung 3.18: Fünf berechnete Basismesskurven des EndpointA-Prozessparameters. Verwendet wurde Architektur 2 ohne PCA-Vorverarbeitung. Rechts unten ist das Konvergenzverhalten des Netzwerkes dargestellt.

Die Untersuchung wird wieder mit und ohne PCA-Vorverarbeitung durchgeführt. Zunächst wird die PCA weggelassen und die Anzahl der Ausgabeneuronen des ICA-Netzwerkes auf fünf reduziert. Um alle Koeffizienten zu berechnen, wurden anschließend 50 Ausgabeneuronen verwendet.
Die entsprechenden Basiskurven sind in Abbildung 3.18 und 3.19 zu sehen. Im Fall von nur fünf berechneten Basiskurven fällt auf, dass die Graphiken starke Oszillationen aufweisen. Dies liegt daran, dass zu wenige Basiskurven bestimmt wurden und deshalb die vielen Merkmale der Originaldaten sich auf wenige Basiskurven verteilen müssen.
Dieses Verhalten ändert sich, falls die Eingabedimension gleich der Ausgabedimension ist und 50 Basiskurven bestimmt werden. In diesem Fall ist das Ergebnis besser interpretierbar und es ergeben sich Basiskurven, die sich auf einzelne Merkmale spezialisieren.

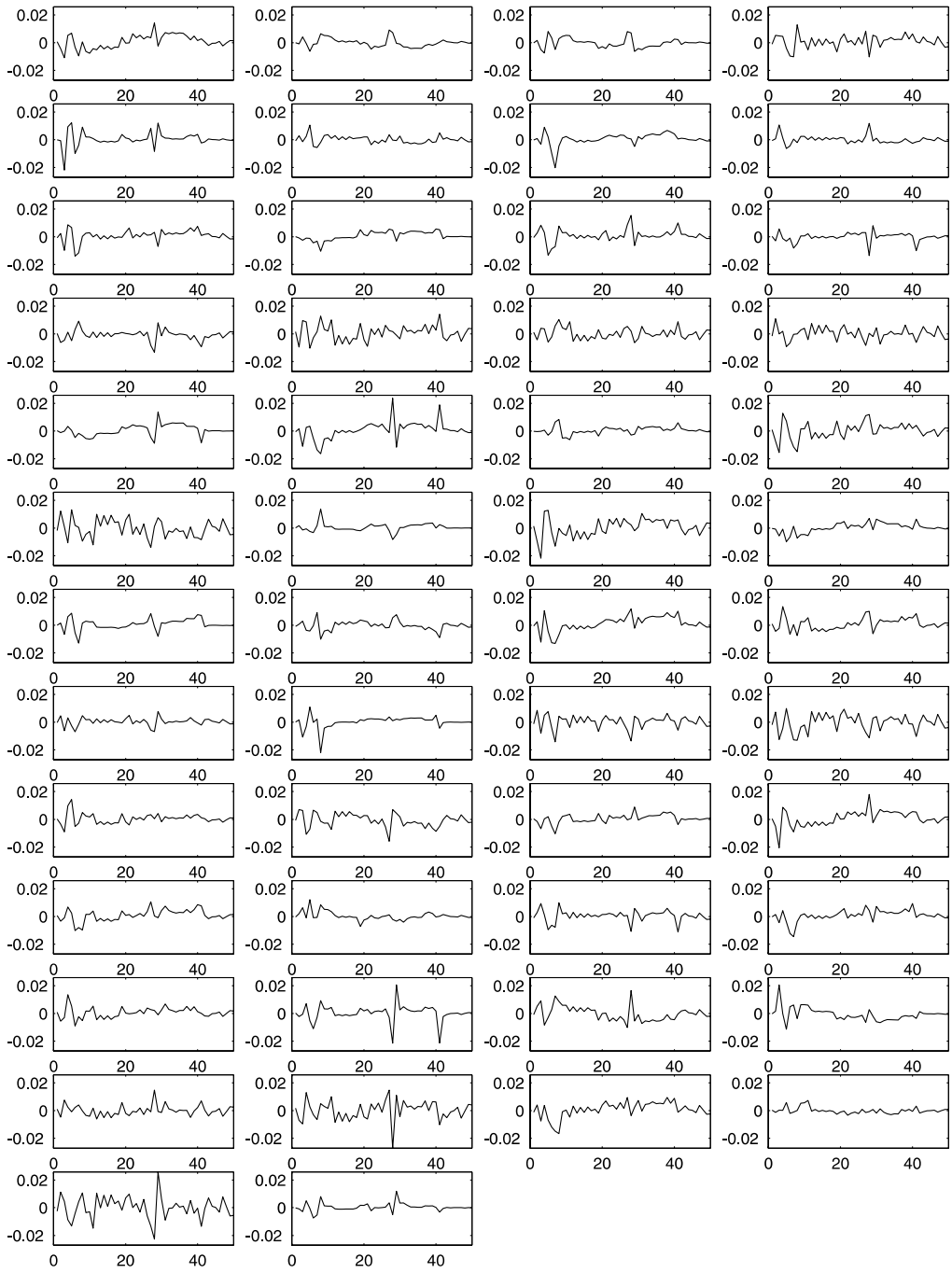

Abbildung 3.19: Fünfzig berechnete Basismesskurven des EndpointA-Prozessparameters mit Architektur 2 ohne PCA-Vorverarbeitung

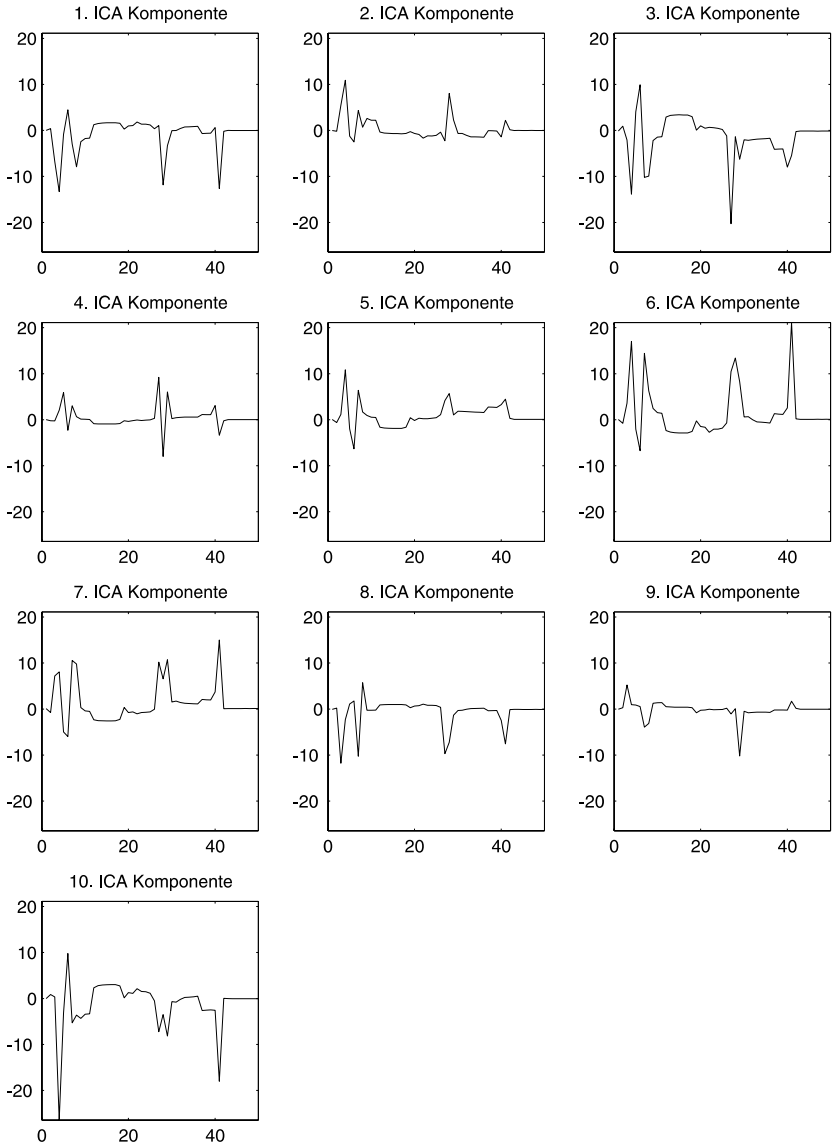

Abbildung 3.20: Zehn berechnete Basismesskurven des EndpointA-Prozessparameters mit Architektur 2 mit PCA-Vorverarbeitung.

Auch bei Architektur 2 wurde der Einfluss einer PCA-Vorverarbeitung auf die ICA-Analyse untersucht. In Abbildung 3.20 sind die zugehörigen Basismesskurven abgebildet. Es wurden 10 Hauptkomponenten, d.h. für eine Dimensionsreduzierung von 50 auf 10 Dimensionen, berechnet. Das anschließend verwendete ICA-Netzwerk besteht aus 10 Eingabe- und 10 Ausgabeneuronen. Die Basisbilder weisen eine große Ähnlichkeit zu den ersten Hauptkomponenten auf, wie der Vergleich mit Abbildung 3.2 zeigt.
Üblicherweise wird die PCA-Analyse zur Dimensionsreduzierung verwendet, da aufgrund der Varianzmaximierung die maximale Information in den verbleibenden Dimensionen vorhanden ist. Eine anschließende ICA-Analyse, ohne Dimensionsreduzierung, kann Korrelationen höherer Ordnung in den Datensätzen lokalisieren und für weitere Analysen indizieren.

Im Fall der untersuchten nichtlinearen Zeitreihen ergeben sich durch die nachgeschaltete ICA-Analyse keine wesentlich neuen Erkenntnisse aus den Daten. In diesem Fall scheint die wesentlich einfachere und schnellere PCA-Vorverarbeitung auszureichen.

Abschließend wird auch für diese Architektur die Rekonstruktionsleistung überprüft. Die rekonstruierten Kurven ergeben sich aus:

$$\boldsymbol{X}^{\boldsymbol{T}}_{rec}=pinv(\boldsymbol{WP})\boldsymbol{U} \tag{3.13}$$

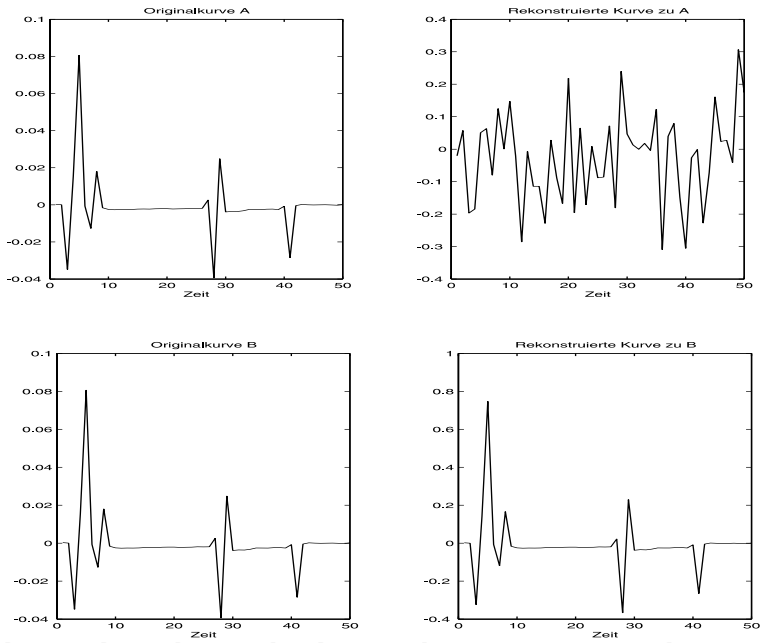

Abbildung 3.21:Rekonstruktion bei Architektur 2 ohne PCA-Vorverarbeitung. Oben: mit fünf Basismesskurven. Unten: mit 50 Basismesskurven. Links ist jeweils die Originalkurve und rechts die jeweilige rekonstruierte Kurve zu sehen.

Wird keine PCA durchgeführt, ist für $\boldsymbol{P}$ die Einheitsmatrix zu setzen. Ohne PCA-Vorverarbeitung ergeben sich die Rekonstruktionen wie in Abbildung 3.21 dargestellt. Bei der Verwendung von nur fünf berechneten Basisbildern ist der Rekonstruktionsfehler sehr groß. Die extrahierten Basismuster enthalten also kaum Information über die Originaldaten. Werden 50 Basismuster extrahiert ist der Rekonstruktionsfehler gleich Null. Dies ist einleuchtend, da in diesem Fall gilt:

$$pinv(\boldsymbol{W})=\boldsymbol{W}^{-1} \tag{3.14}$$

$$\Rightarrow \boldsymbol{X}_{rec}= pinv(\boldsymbol{W})\boldsymbol{U} = \boldsymbol{W}^{-1}\boldsymbol{W}\boldsymbol{X} = \boldsymbol{X} \tag{3.15}$$

Wird eine PCA als Vorverarbeitungsschritt durchgeführt, werden für eine gute Rekonstruktion weniger Basismuster benötigt. In Abbildung 3.22 ist die Rekonstruktion mit 10 Basismustern nach vorgeschalteter PCA-Analyse zu sehen.

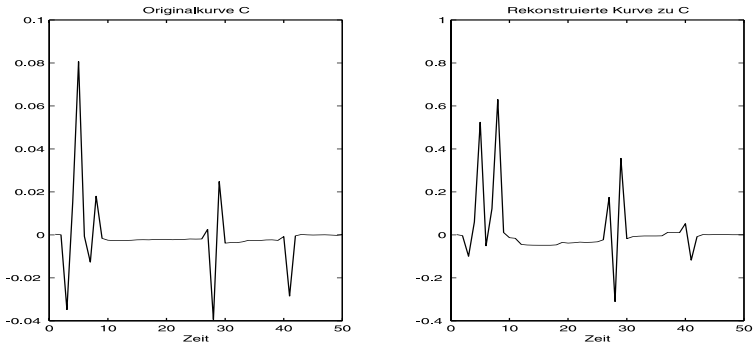

Abbildung 3.22:Rekonstruktion bei Architektur 2 mit PCA-Vorverarbeitung wobei 10 Basismesskurven verwendet wurden. Links ist die Originalkurve und rechts die rekonstruierte Kurve dargestellt.

Wie bereits erwähnt, ist die Qualität der Rekonstruktion für die ICA nicht das entscheidende Kriterium. Ziel der ICA-Analyse ist, möglichst statistisch unabhängige Ausgabewerte zu erhalten. Bei Architektur 1 war dies für das untersuchte Datenmaterial nicht der Fall. Für die Architektur 2 wurde die analoge Berechnung durchgeführt. Die Ergebnisse sind in Tabelle 3.9 dargestellt. Man sieht sehr deutlich, dass der ICA-Algorithmus bei Architektur 2 stets eine Ausgabe liefert, die statistisch unabhängiger ist, als die Ausgabe des entsprechenden PCA-Algorithmus.

Tabelle 3.9: Mutual Information bei Architektur 2

	Ausgabeneuronen	M
ICA	5	0.317
PCA	5	0.341
ICA	50	0.026
PCA	50	0.246

Bei fünf Ausgabeneuronen muss man jedoch mit dieser Aussage vorsichtig sein, da die Unterschiede der „*Mutual Information*“ M durch statistische Schwankungen erklärbar sein könnten. Bei 50 Ausgabeneuronen unterscheiden sich die Werte um eine Zehnerpotenz, so dass hier davon ausgegangen werden kann, dass der ICA-Algorithmus tatsächlich die statistisch unabhängigeren Ausgabewerte im Vergleich zur PCA liefert.
In diesem Beispiel sind die Bedingungen erfüllt, die bereits bei Architektur 1 erwähnt und dort nicht erfüllt waren. Die Eingabedimensionalität entspricht der Ausgabedimensionalität und die statistische Basis ist aufgrund der 1695 Werte besser gegeben.

3.1.5 Multivariate Datenanalyse

In diesem Abschnitt wird kurz die multivariate Datenanalyse und Klassifikation der multisensorischen nichtlinearen Zeitreihen behandelt. Bei den bisher beschriebenen Methoden standen hauptsächlich die Dimensionsreduzierung unter Varianz- und damit Informationsmaximierung bzw. Bestimmung von statistisch unabhängigen und damit unkorrelierten Informationen einer einzelnen Zeitreihe im Vordergrund.

Multivariate Datenklassifikation ist immer ein komplexes Thema, falls die Dimensionalität des Problems größer als drei ist und damit nicht mehr einfach visualisiert werden kann. Zur unüberwachten Datenklassifikation wurde von verschiedenen Autoren in den letzten Jahren eine Vielzahl unterschiedlicher Algorithmen vorgeschlagen. Adaptive-Resonanz-Theorie-Netze (ART) z.B. wurde von Grossberg [Grossberg76, 76a] als clusterbasiertes autonomes Lernmodell entwickelt. Die NASA entwickelte *unsupervised* Bayesian Klassifikationsschemata [Cheeseman96, 88, 89]. Kohonennetze wurden in dieser Arbeit bereits untersucht, um nur eine kleine Auswahl zu nennen.

Ein grundsätzliches Problem der unüberwachten Klassifikation stellt die Entscheidung dar, wann keine neue Klasse mehr zu bilden ist [Everitt81]. Eine zu große Anzahl von Klassen, beschreibt in der Regel den Phasenraum zu detailliert, so dass die Anzahl der Klassen beschränkt werden muss. Im Fall von neuronalen Netzen spricht man dabei auch vom Problem des *Overfitting*. Manche Systeme benutzen ein *ad-hoc*-stop-Kriterium. So wird bei ART2 ein Schwellenwert gesetzt, der vom Anwender über *trial and error* ermittelt werden muss. Bei Kohonennetzen muss die Anzahl der Klassen bereits beim Netzwerkdesign berücksichtigt werden.

Prinzipiell sollte nur dann eine multivariate Auswertung bzw. Klassifikation durchgeführt werden, wenn die zugrundeliegenden Daten dies unbedingt erfordern. Die zu untersuchende Datenmenge wurde aus diesem Grund auf multivariate Effekte untersucht. Auf weiterführende Untersuchungen zur multivariaten Klassifikation sei auf das Kapitel 3.3 zur Mustererkennung und Klassifizierung hochdimensionaler Testdatenfelder verwiesen.

Bereits eine einfache Graphik der ersten beiden Hauptkomponenten der *EndpointA*-Kurven und Chiller-Bath-Temperatur zeigen komplexe Strukturen im zweidimensionalen Raum. Eine anschließende ICA-Analyse sollte durch statistische Unabhängigkeit eine evtl. einfachere Struktur im Phasenraum erreichen. In Abbildung 3.23 ist sowohl der ursprüngliche Eingaberaum, der transformierte ICA-Ausgaberaum, sowie die anschließende Kohonenkarte zu sehen, die bereits in Kapitel 3.1.3 untersucht wurde.

Das ICA-Netzwerk dreht die Eingabedaten um 90 Grad. Dies war bei jeder anderen Paarung von Datensätzen auch feststellbar. Es ergeben sich bereits im zweidimensionalen Raum komplexe Strukturen, die eine multivariate Betrachtung für die Klassifikation erfordern.
Die Suche nach Ausreißern gestaltet sich dabei, wie oben kurz skizziert, natürlich schwierig und es könnten Algorithmen, wie in Kapitel 3.1.3 angesprochen, verwendet werden.

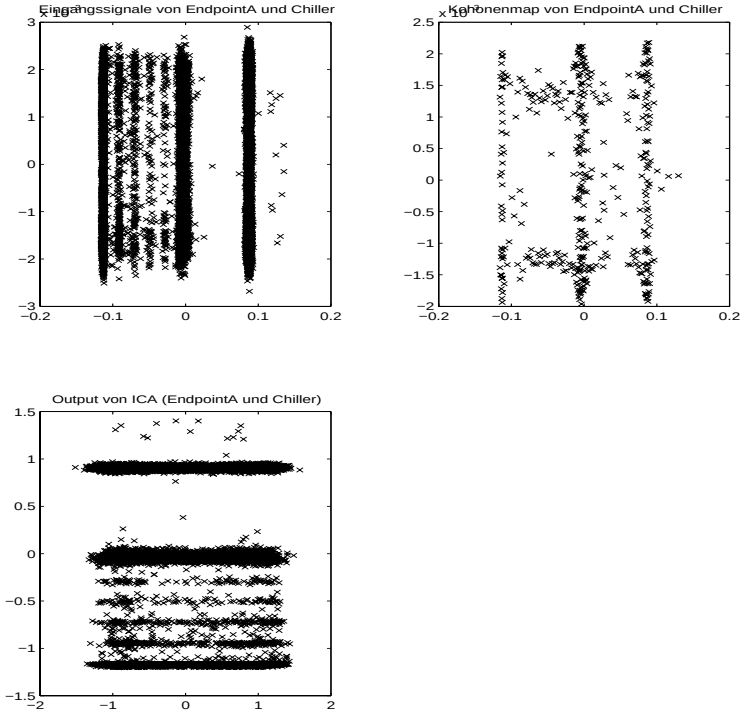

Abbildung 3.23: Multivariate Darstellung von Eingabedaten. Bereits im zweidimensionalen Raum ergeben sich komplexe Strukturen im Phasenraum. Links oben: Trainingsvektoren von EndpointA und Chiller-Bath-Temperatur im Eingaberaum. Rechts oben: Kohonenkarte des Eingaberaumes. Links unten: Ausgabe des konvergierten ICA-Netzwerkes. Im wesentlichen ergibt sich eine Drehung des Eingaberaumes, die bereits bei mehreren ICA-Analysen festgestellt wurde.

Aufgrund der fehlenden Information über das zu den Zeitreihen entsprechende Prozessergebnisses, ist es mit den untersuchten Daten praktisch nicht möglich, die Klassifikationsergebnisse auf deren Relevanz zu überprüfen.

Die prinzipiellen Analysen in diesem Kapitel zeigen, dass der Phasenraum multivariate Effekte zeigt, die im Fall einer automatischen Klassifikation berücksichtigt werden müssen. Für die Untersuchungen weiterer Daten standen zum Teil auch die Prozessergebnisse zur Verfügung. Die Untersuchungen von nichtlinearen multisensorischen Zeitreihen mit Vorwissen wird im nächsten Kapitel, die Untersuchung von multivariaten Phasenräumen mit und ohne Vorwissen im Kapitel 3.3 betrachtet.

3.1.6 Diskussion

Ziel der Untersuchungen in diesem Kapitel war es, aktuelle Analysemethoden mit neuronalen Netzen zur Auswertung multisensorischer nichtlinearer Zeitreihen zu verwenden und bzgl. des Einsatzes zur Datenvorverarbeitung, Dimensionsreduzierung und Kennzahlenextraktion zu bewerten. Als Testdatensätze standen dabei Zeitreihen von Prozessvariablen zur Verfügung, die während der Bearbeitung von Silizium-Scheiben mit einer Samplingrate von 1 Hz aufgezeichnet wurden. Die Prozessierungsdauer variierte zwischen 280 und 500 Sekunden, Datensätze waren von 1695 Prozessierungen verfügbar und pro Wafer wurden 10 Prozessvariablen aufgezeichnet. Zu diesen Zeitreihen standen jedoch keine Aussagen über das Prozessergebnis, oder eine andere *a-priori*-Klasseneinteilung zur Verfügung.
Nach einer Datenvorverarbeitung wurden PCA (Principal Component Analysis) als Methode zur Dimensionsreduzierung und Kennzahlenextraktion, ICA (Independent Component Analysis) zur Erzeugung statistisch unabhängiger Information und Kohonennetze zur Klassifikation verwendet.

Zuerst wurden die Messdaten mittels einer einfachen Mittelwertbildung auf einheitlich 50 Datenpunkte skaliert und anschließend auf die Länge Eins normiert. Die anschließend durchgeführte PCA zeigte, dass in den ersten beiden Hauptkomponenten nahezu die gesamte Varianz bzw. Information über die Messkurven enthalten ist. Als qualitatives Maß wurde der mittlere quadratische Rekonstruktionsfehler verwendet. Die Methode der PCA wurde ursprünglich um die Jahrhundertwende entwickelt [Pearson01, Hotelling33] und vor einigen Jahren als wichtige Technik der Datenanalyse wiederentdeckt. Die zentrale Idee besteht darin, den Datenraum so zu transformieren, dass unter Vernachlässigung von einzelnen Dimensionen die maximale Variation in den verbleibenden Dimensionen enthalten bleibt.

Ein grundsätzliches Problem der unüberwachten Klassifikation stellt die Entscheidung dar, wann keine neue Klasse mehr zu bilden ist [Everitt81]. Eine zu große Anzahl von Klassen beschreibt in der Regel den Phasenraum zu detailliert, so dass die Anzahl der Klassen beschränkt werden muss. Manche Systeme benutzen ein *ad-hoc*-stop-Kriterium. So wird bei ART2 ein Schwellenwert gesetzt, der vom Anwender über *trial and error* ermittelt werden muss.

Als Clusterverfahren wurde, in diesem Fall des unüberwachten Lernens, die Methode der Kohonennetze eingesetzt. Da die meiste Information in zwei Hauptkomponenten enthalten ist, wurde ein zweidimensionales Kohonennetzwerk benutzt. Neben dem Geschwindigkeitsvorteil beim Lernen bzw. in der Recallphase können zweidimensionale Netze gut visualisiert werden. Für die *EndpointA*-Datensätze ergaben sich deutliche Cluster im zweidimensionalen Raum und einige Ausreißer, d.h. Gebiete mit niedriger Population im Phasenraum. Diese Neuronen können zur Detektion von Ausreißern bzw. abnormalen Gebieten verwendet werden. Eine gut- bzw. schlecht-Klassifikation, die für den produktiven Einsatz von Vorteil wäre, kann aufgrund dieser Bewertung nicht durchgeführt werden. Der Rückschluss von abnormal zu

schlechtem Prozessergebnis hängt stark von den Prozesstoleranzen ab und kann nur in der Analyse bzw. Korrelation der Zeitreihe mit dem jeweiligen Prozessergebnis mit signifikanter Wahrscheinlichkeit angegeben werden.
Für die Betrachtung in diesem Kapitel reichte es allerdings schon, eine normal/abnormal Bewertung zu treffen, da im Fall einer Normalprozessierung eine große Wahrscheinlichkeit auch eine gute Prozessierung, im Sinne von Erreichung des Prozesszieles, vorliegt. Bei über 400 Prozessierungsschritten müssen die Normalgebiete auf eine gute Prozessierung schließen lassen, sonst könnte die Ausbeute von derzeit über 90% nicht erreicht werden.
Konkret für das Beispiel von 90% Ausbeute muss die Wahrscheinlichkeit für eine gute Prozessierung pro Prozessschritt bei $0{,}9^{1/400} = 0{,}997$ liegen. Anders formuliert heißt das, dass für die Trockenätzungen, die in diesem Kapitel untersucht wurden, im statistischen Mittel eine Fehlerrate kleiner als 0,3% vorliegen muss, was bei der vorliegenden Datenmenge von 1695 Wafer ca. 5 Wafer entspricht.

Phasenraumbereiche niedriger Dichte können also als kritische Bereiche betrachtet werden - sei es als qualitätsrelevant oder auch produktivitätsrelevant, im Sinne von ungeplanten nicht produktiven Stillständen der Anlage. So müssen üblicherweise die Prozessanlagen in zyklischen Abständen gewartet werden. Eine Prozesscharakterisierung durch die Prozessparameter in Phasenräume niedriger Dichte weisen auf bevorstehende Probleme von Anlagenteilen wie z.B. Lecks in den Vakuumkammern hin und sind trotz der nicht eindeutig zuordenbaren gut/schlecht Klassifizierung von Interesse.

Für die Auswertung der Kohonenkarten wurden verschiedene Algorithmen angewendet. Prinzipielle Überlegung ist, dass die Kohonenkarte die Wahrscheinlichkeitsdichte des Eingaberaumes darstellt und Ausreißer zu einer geringen lokalen Neuronendichte in der Kohonenkarte führen. Die einfachste Methode der Ausreißerdetektion besteht in der univariaten Betrachtung der einzelnen Dimensionen mit Methoden der statistischen Prozesskontrolle, wie Mittelwert +/- 3 Sigma. Werte innerhalb dieses Bereiches stellen die Normalgebiete, Werte außerhalb die Ausreißer bzw. abnormalen Phasenraumgebiete dar.
Die multivariate Betrachtung der Hauptkomponenten der Prozessparameter zeigt komplexe Strukturen im mehrdimensionalen Raum, was auf ein hochdimensionales Klassifikationsproblem hinweist. Aus diesem Grund wurden zwei Methoden zur Ausreißerbestimmung verwendet, die sowohl im mehrdimensionalen Gültigkeit besitzen, als auch eine Analysegeschwindigkeit zulassen, die eine *in situ* Analyse ermöglichen.

Eine Methode bestand in der Bestimmung der lokalen Neuronendichte durch Rasterung der Kohonenkarte in n-dimensionale Quader. Dieses Vorgehen erinnert sehr stark an RCE-Klassifikationsalgorithmen (*Reduced Coulomb Energy*) [Reilly82, Moreno95]. Die Problematik bei diesem Vorgehen besteht in der Abhängigkeit des Ergebnisses von der Präsentationsreihenfolge der Trainingsmuster, der Bestimmung der Initial-Kantenlänge der n-dimensionalen Würfel und der Festsetzung des Dichtegrenzwertes zur Klasseneinteilung normal/abnormal. Qualitativ kann zumindest die Aussage getroffen werden, dass die Dichteverteilung in den Phasenraumsegmenten indirekt proportional zur Wahrscheinlichkeit ist, dass diese Neuronen abnormale Wafer detektieren.

Als zweite Methode zur Auswertung der Kohonenkarten wurde die Nächste-Nachbar-Methode angewendet. Als Separationskriterum zur Klasseneinteilung dient hier die Bestimmung des mittleren Abstandes zu allen anderen Neuronen. Auch dieser Algorithmus stellt eine Methode zur Bestimmung der lokalen Neuronendichte dar, und auch hier muss wieder ein *ad-hoc*-Grenzwert als Schwellenwert definiert werden.

Die Auswertungen der *EndpointA*-Zeitreihen ergab, dass für Ausreißer überwiegend gestauchte Kurven, d.h. kürzere Prozessierungen als normal, und unterschiedliche Plateauhöhen im *Mainetch* als Ursache für die Einteilung als abnormal verantwortlich sind. Die Detektion von zu kurzen Prozessierungsdauern ist dabei vollkommen richtig, da üblicherweise die Ätzrate, d.h. die Ätzgeschwindigkeit, einen konstanten Wert besitzt bzw. besitzen soll. Ist die Prozessierung kürzer als üblich, ist entweder die Ätzrate höher (was wiederum an fehlerhaften Gaszuflüssen oder zu hoher eingestrahlter Leistung liegen kann) oder es befinden sich noch Reste der abzutragenden Schicht auf der Scheibe. Die unterschiedlichen Plateauhöhen des *EndpointA*-Signals können unterschiedliche Ursachen haben. Zum einen kann sich tatsächlich die Chemie im Reaktor geändert haben, so dass sich das Emissionsspektrum und damit auch die Intensität der erfassten Wellenlänge ändert. Zum anderen kann allerdings auch ein einfacher Operatoreingriff (die erfassten Werte sind relative Werte, die manuell justierbar sind) diese Abnormität hervorrufen. Für unsere Untersuchungen sind die prinzipiellen Ursachen jedoch zweitrangig. Von primärem Interesse ist, dass diese Scheiben durch abnormale Signalverläufe charakterisiert sind und als abnormal von den Analyse-Algorithmen detektiert werden.

Andere Datensätze, wie z.B. Chiller-Bath-Temperatur, zeigten ein abweichendes Verhalten. Es entstand ein symmetrisches Kohonennetz ohne erkennbare Ausreißer. Diese Beobachtung ist insofern interessant, da dies indiziert, dass die Ursachen für die Abnormitäten nicht in allen Prozessparametern ersichtlich sind. Zur Detektion von relevanten Abnormitäten bzw. im zweiten Schritt in der erfolgreichen Korrelation von Prozessparametern mit Prozessergebnissen, ist also eine Voraussetzung, die richtigen Prozessvariablen überhaupt zu erfassen. Dies ist insofern nicht trivial, da diese Prozessanlagen mehrere hundert Parameter prinzipiell erfassen könnten, aufgrund der Beschränkung der Schnittstelle auf 9600 Bd allerdings nur wenige tatsächlich erfassen. Die erste Herausforderung in der erfolgreichen Analyse besteht also in der richtigen Auswahl von Prozessparametern. Ein Punkt dabei ist, dass möglichst Parameter mit geringer Kreuzkorrelation und darüber hinaus natürlich die relevanten Prozessparameter, d.h. Parameter, bei denen sich schwerwiegende Fehler in Änderungen der Zeitreihen bemerkbar machen, erfasst werden. Zur Detektion relevanter Prozessparameter wird im nächsten Kapitel eine Methode, basierend auf Multilayer-Perzeptrons mit Backpropagation-Lernregel, vorgestellt.

Die Zahl der Hauptkomponenten wurde schließlich von Zwei auf Fünf erhöht. Damit ergibt sich eine fünfdimensionale Kohonenkarte, bei der die jeweiligen zweidimensionalen Schnitte näher untersucht wurden. Die Schwierigkeit dabei ist, interessante Schnitte bzw. interessante Projektionen der Karte zu finden. Neueste Untersuchungen verwenden hierzu Exploratory Projection Pursuit-Verfahren (EPP) [Huber85].

Die sogenannte Independent Component Analysis (ICA) wurde anschließend verwendet, um statistisch unabhängige Komponenten zu finden und so zu einer weiteren Dimensionsreduzierung und Informationsmaximierung zu gelangen. Im wesentlichen wurden zwei verschiedene Architekturen [Bartlett98] mit unterschiedlichen Eingabe- und Ausgaberäumen verwendet. Die Unterschiede bestehen dabei in der unterschiedlichen Interpretation des Ergebnisses:
Bei Architektur 1 wurde als Eingabevektor der gleiche Datenpunkt von allen Messkurven verwendet. Mit diesen Vektoren wurde mit bzw. ohne PCA-Vorverarbeitung ein ICA-Netzwerk trainiert. Als Ausgabe erwartet man statistisch unabhängige Basismesskurven.
Architektur 2 verwendet als Eingabe eine ganze Messkurve. Auch bei dieser Architektur wurde mit bzw. ohne PCA als Vorverarbeitungsschritt das ICA-Netzwerk trainiert. Als Ergebnis erwartet man statistisch unabhängige Koeffizienten. Dieses Vorgehen wird in der Literatur auch als Faktorencode bezeichnet.

Um eine ICA durchzuführen, gibt es verschiedene Ansätze [Girolami99]. In dieser Arbeit wurde das Prinzip der MMI verwendet, d.h. die „*Mutual Information*“ der Ausgabe wurde minimiert. Für beide Architekturen wurden Basismesskurven berechnet, die bei Architektur 1 statistisch unabhängig sein sollten. Als Ergebnis erhält man Basismesskurven, die für Architektur 1 eine auffällige Übereinstimmung mit den Hauptachsen der PCA aufweisen und deshalb auch eine relativ gute Rekonstruktionsleistung erreichten. Bei der ICA mit der Architektur 2 ergeben sich komplexere Strukturen. Die Rekonstruktionsleistung war nicht sehr gut und nur im Fall hoher Dimensionalität des Ausgaberaumes, der ähnlich der Eingabedimensionalität ist, wird eine zufriedenstellende Rekonstruktionsleistung erbracht.
Das Ziel einer ICA ist allerdings nicht die Informationsmaximierung bei einer Dimensionsreduzierung, sondern die Bestimmung statistisch unabhängiger Information. Das ungewöhnliche Ergebnis bei der Untersuchung von Architektur 1, die PCA-Hauptkomponenten waren statistisch unabhängiger als die ICA-Basismesskurven, lässt sich durch die unterschiedliche Dimensionalitäten der Eingabe- und Ausgaberäume und die zu geringe Anzahl von Datenpunkten erklären. Bei Architektur 2 waren diese Bedingungen gegeben und man beobachtet im Vergleich zur PCA-Ausgabe stets eine statistisch unabhängigere Ausgabe durch die ICA-Analyse.

3.2 Methoden zur Kennzahlenextraktion, Klassifizierung und Analyse nichtlinearer Zeitreihen mit Vorwissen

Dieses Kapitel behandelt die Untersuchung von Methoden zur Kennzahlenextraktion, Klassifizierung und Analyse nichtlinearer multisensorischer Zeitreihen, wie sie während der Prozessierung von Si-Wafer aufgezeichnet werden.
Zum verwendeten Datenmaterial war, im Gegensatz zu den Zeitreihen aus Kapitel 3.1, die Information vorhanden, dass gewisse Merkmale auf den Zeitreihen eine schlechte Prozessierung indizieren. Nach der Vorverarbeitung und Analyse mit Verfahren der nichtlinearen Zeitreihenanalyse werden Backpropagation-Netze zur Kennzahlenextraktion und Parameterselektion, Kohonen Self Organizing Feature Maps (SOM) und Self Organizing Surfaces (SOS) zur automatischen Klasseneinteilung und RBF-Netzwerke (Radial Basis Function) zur Datenklassifikation verwendet. Im letzen Abschnitt dieses Kapitels werden die Ergebnisse diskutiert.

3.2.1 Datenvoranalyse

In diesem Abschnitt werden durch Methoden der nichtlinearen Zeitreihenanalyse Parameter für die Modellierung der zugrundeliegenden Dynamik des Systems bestimmt. Damit wird eine Abschätzung der Komplexität des Problems bzw. eine Abschätzung der Netzwerkarchitektur zur Systemmodellierung ermöglicht.
Erster Schritt bei der Analyse einer neuen Zeitreihe ist auch hier, sich einen qualitativen Überblick über die Beschaffenheit der Daten und damit über das zugrundeliegende System zu besorgen. Die Visualisierung des Signalverlaufs gegen die Zeitreihe vermittelt einen ersten Eindruck, da Informationen wie Nichtlinearitäten oder Periodizitäten oftmals unmittelbar erkennbar sind.

Eine weitere wichtige Eigenschaft, die einen wesentlichen Einfluss auf die Art der Datenanalyse hat, ist die Dynamik des zugrundeliegenden Systems. In vielen Fällen kann durch die Datenvoranalyse festgestellt werden, ob das System (i) linear oder nichtlinear, (ii) deterministisch oder stochastisch, (iii) regulär oder chaotisch ist. An dieser Stelle ist v.a. der Unterschied zwischen deterministisch chaotisch und stochastischen Systemen wichtig. Erstere können dynamisch modelliert, bei letzteren können nur Wahrscheinlichkeitsaussagen getroffen werden.

Eine Zeitreihe $\{x(t):t=1,2,...,N\}$ wird normalerweise durch die Messung einer dynamischen skalaren Variablen in einem konstanten Zeitintervall Δt bestimmt. Die Samplingrate ergibt

sich aus der Optimierung zwischen Redundanz und völliger Unkorreliertheit aufeinanderfolgender Datenpunkte in der Zeitreihe. Eine unpassende Samplingrate zeigt oftmals eine falsche Dynamik und führt zu falschen Analyseergebnissen. Prinzipiell wird die Samplingrate von der Komplexität bzw. dem chaotischen Verhalten beeinflusst. Je höher das chaotische Verhalten, desto größer sollte auch die Samplingrate sein, damit die Dynamik des zugrundeliegenden Systems ausreichend in der Zeitreihe enthalten ist. Nach dem Nyquist-Theorem muss die Samplingrate dabei doppelt so groß sein, wie die maximale Frequenz des Signals, damit das Signal eindeutig rekonstruiert werden kann [Press98].

Viele Analysemethoden setzen eine ausreichend lange Messzeit voraus, um die Dynamik vollständig in den Daten wiederzufinden, und um die Stationarität der Daten zu erreichen. Die Modellierung des Lufttemperaturverlaufs wird z.B. nur dann funktionieren, wenn Daten eines ganzen Jahres verfügbar sind. Die zugrundeliegende Dynamik lässt sich dabei in kurzfristige und langfristige Dynamik unterteilen.

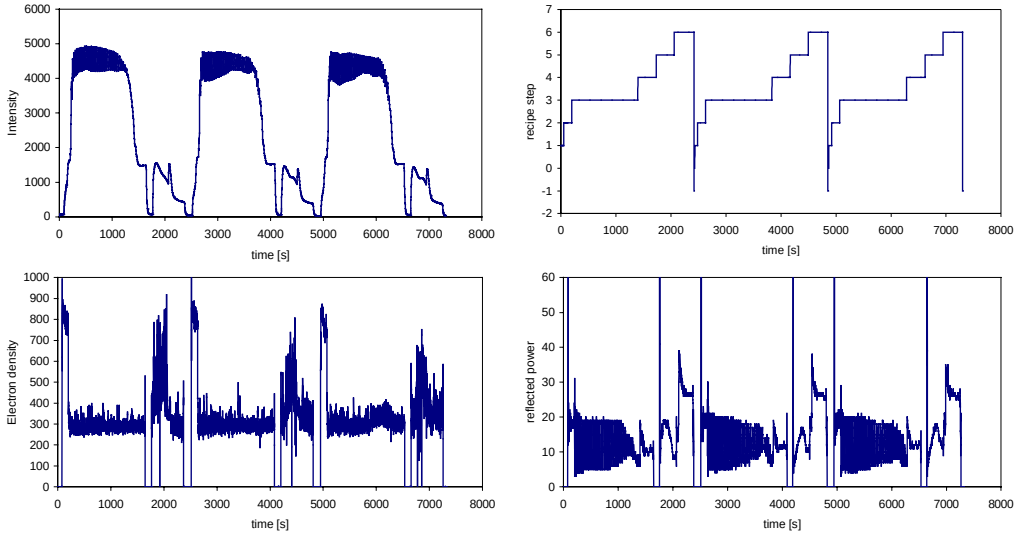

Abbildung 3.24: Original Sensor Signalverläufe einer Amat 8300 Alu Hex Plasmaätzanlage. Gezeigt sind drei typische Signalverläufe die während der Ätzung erfasst werden. Gezeigt sind das Emissionsspektrum des Plasmas bei einer Wellenlänge von 396 nm, die reflektierte Leistung und die Elektronendichte. Rechts oben ist der Recipe-Step dargestellt.

In Abbildung 3.24 sind die original Sensor Signalverläufe einer Amat 8300 Alu Hex Trockenätzanlage gezeigt. Es werden bis zu 50 zeitabhängige Sensorsignale, von internen und nachträglich installierten externen Sensoren, parallel, aufgezeichnet. Der Parameter *Recipe-Step* wird zu den Signalen gezeichnet, da abhängig vom *Recipe-Step* unterschiedliche Rezepteinstellungen aktiv werden. Im *Mainetch* (*Step* 3) z.B. wird mit Cl Aluminium geätzt. Die abfallende Flanke des Emision_1 Signals deutet darauf hin, dass die zu ätzende Alu-Schicht weggeätzt ist. Beim sogenannten *Lackstrippen* (*Step* 5) wird mit Sauerstoff und erhöhtem Druck der Lack „verascht“.

Einen ersten Überblick über die Autokorrelation der Zeitreihe erhält man durch die Visualisierung aufeinanderfolgender Datenpunkte, indem einfach die Punkte x_i gegen $x_{i-\Delta t}$ gegeneinander aufgetragen werden. Ist die Verteilung der Punkte sehr nah an der Diagonalen

ist die Samplingrate, und damit die Redundanz, der Zeitreihe ausreichend hoch und die Dynamik des Systems kann sehr wahrscheinlich aus der Datenreihe rekonstruiert werden [Kulkarni97].

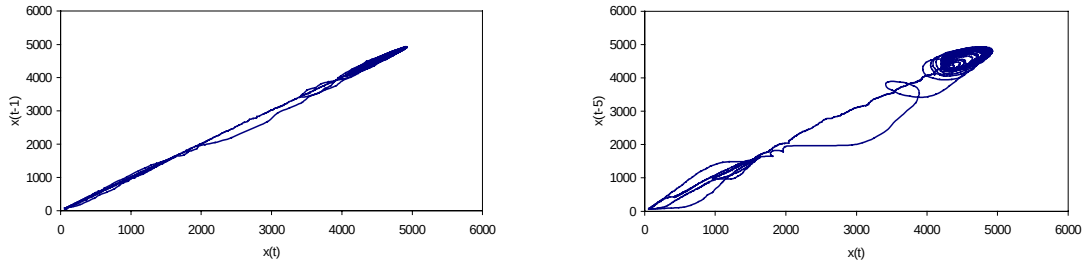

Abbildung3.25: Autokorrelationsdiagram der Zeitreihe eines Runs mit verschiedenen Zeitverzögerungen. Bei einer Samplingrate von 1 Hz zeigt das Diagramm für die Zeitverzögerung von Δt=1 eine starke Autokorrelation. Erst ab einer Zeitverzögerung von Δt=5 nimmt die Redundanz deutlich ab.

Das Autokorrelationsdiagramm des Emissionsspektrums in Abbildung 3.25 zeigt, dass die Samplingrate, zumindest für diesen Parameter, ausreichend hoch gewählt wurde und die Dynamik des zugrundeliegenden Systems in den Daten ausreichend enthalten sein müsste.

Durch die Darstellung des Kreuzkorrelationsdiagramms erkennt man Abhängigkeiten zwischen den Parametern. Im vorherigen Kapitel über unüberwachte Methoden zur Datenanalyse wurde festgestellt, dass, zumindest für die aufgetretenen Abnormitäten, sich Anlagen- und Prozessprobleme nicht in allen Prozessvariablen bemerkbar machen. Für die Kennzahlenextraktion ohne Vorwissen sollten deshalb alle Prozessvariablen berücksichtigt werden. In diesem Kapitel, ist zumindest für ein Fehlerbild, eine Klasseneinteilung bekannt, so dass diese Ergebnisse verifiziert werden können.

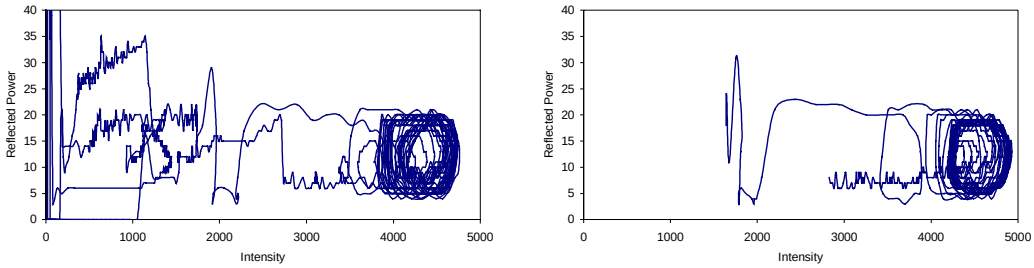

Abbildung 3.26: Wellenlängenintensität (Emision_1) und reflektierte Leistung. Bei der gleichzeitigen Visualisierung dieser Signalverläufe ist die Kreuzkorrelation zwischen diesen Sensorsignalen deutlich zu erkennen. Links: Diagramm über die gesamte Prozessierungsdauer. Rechts: Kreuzkorrelationsdiagramm während des Mainetch-Steps. Bereits im zweidimensionalen Raum ergeben sich komplexe Strukturen.

Im Folgenden werden die beschriebenen Methoden der nichtlinearen Zeitreihenanalyse auf eine Observable angewendet, um einen Eindruck von der Komplexität bzw. Dimensionalität des Problems und für die Abschätzung des Netzwerkdesigns zu erhalten.
Die AMI gibt die mittlere Information über s_n an, wenn $s_{n-\tau}$ bekannt ist, wobei τ die Zeitverzögerung aufeinanderfolgender Datenpunkte darstellt. Ist s_n statistisch unabhängig von

$s_{n-\tau}$, dann ist $S(\tau)$=0. Für $\tau=0$ gibt die AMI die Information an, die durch die Messung von s_n gewonnen wird und entspricht der Shannon Entropie.

Abhängig ist die AMI vom Informationsgehalt der Zeitreihe und kann als Maß für die Chaotik der Zeitreihe verwendet werden. Je chaotischer die Zeitreihe, desto größer ist die durch die Messung gewonnene Information. Eine weitere Abhängigkeit ergibt sich durch das Samplingintervall. Ein großes Samplingintervall führt zu einer weniger starken Redundanz aufeinanderfolgender Datenpunkte und damit zu einem größeren Informationsgewinn durch die Messung.

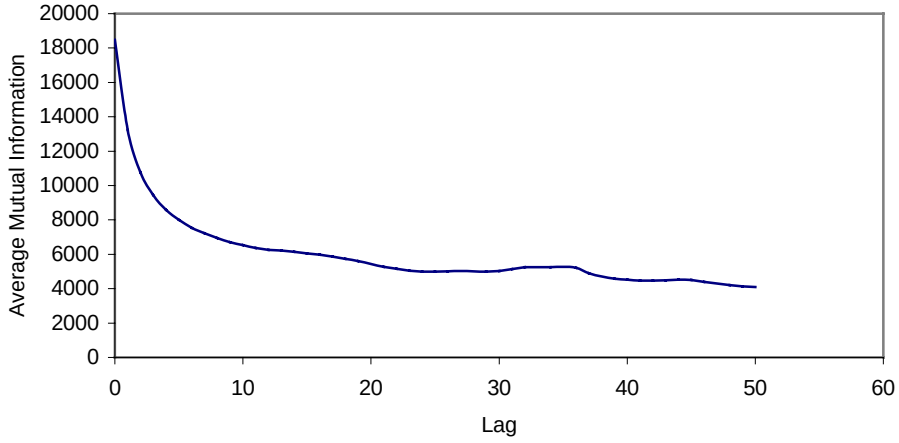

Abbildung 3.27: Average Mutual Information in Abhängigkeit vom Zeitversatz zur Bestimmung der Einbettungsverzögerung τ_E. Das erste Minimum indiziert einen geeigneten Wert für τ_E. Das erste Minimum ist bei einem Lag von 13. Da die Samplingrate der Originalzeitreihe 1 Hz beträgt, entspricht dies einem τ_E von 13 sec.

Die starke Korrelation aufeinanderfolgender Werte der Zeitreihe zeigt, dass die Samplingrate, zumindest im statistischen Mittel, ausreichend hoch gewählt wurde. Das erste Minimum in Abbildung 3.27 gibt einen passenden Wert für die Einbettungsverzögerung an. Diese Information ist neben der geschätzten Attraktordimension ein wichtiger Parameter zur Attraktor-Rekonstruktion. Die Verwendung einer passenden Einbettungsverzögerung kann dabei auch als nachträgliche Korrektur einer nicht optimalen Samplingrate betrachtet werden.

In Abbildung 3.28 ist der rekonstruierte Attraktor aus der *Emission_1*-Zeitreihe dargestellt. Als Einbettungsdimension wurde das durch die *Average Mutual Information* bestimmte τ_E = 13 sec verwendet. Deutlich zu erkennen sind Bereiche im Phasenraum, die Vergleiche mit einem zyklischen Lorenzattraktor nahe legen.
Zur Bestimmung der Attraktordimension wurde mit der *false-nearest-neighbor*-Methode (fnn) gearbeitet. Die passende Dimension ist erreicht, wenn keine, oder zumindest nur noch wenige, falsche nächste Nachbarn vorhanden sind. Der untersuchten Zeitreihe liegt mindestens ein dreidimensionaler Attraktor, höchstens ein sechsdimensionaler Attraktor zugrunde. Da die Definition eines falschen nächsten Nachbarn nicht exakt determiniert ist und die Anzahl der fnn's langsam gegen Null konvergiert, kann in diesem Fall nur ein Bereich für die zugrundeliegende Attraktordimension angegeben werden. Auch für höhere Dimensionen

bleibt die Anzahl der fnn's bei Null, damit kann davon ausgegangen werden, dass es sich hierbei um ein deterministisches System ohne statistisches oder farbiges Rauschen handelt.

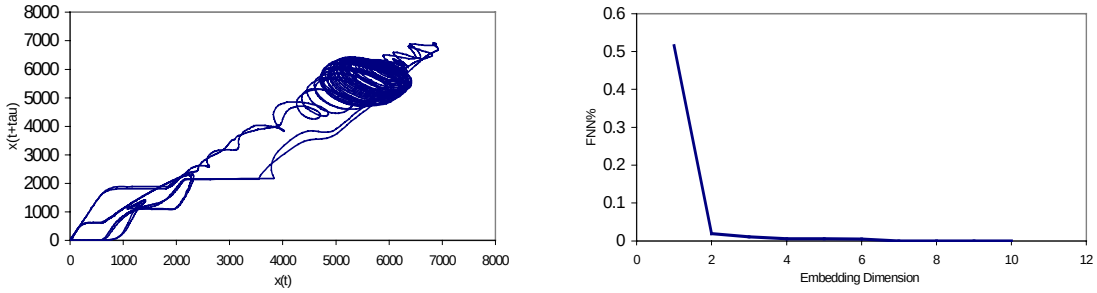

Abbildung 3.28: Links: rekonstruierter Attraktor aus der Zeitreihe der Wellenlängen-intensität. Als Einbettungsdimension wurde das durch die Average Mutual Information bestimmte τ_E = 13 sec verwendet. Deutlich zu erkennen sind Bereiche im Phasenraum die Vergleiche mit einem zyklischen Lorenzattraktor nahe legen. Rechts: Bestimmung der Attraktordimension mit der false-nearest-neighbor-Methode. Die passende Dimension ist erreicht, wenn keine, oder zumindest nur noch wenige, falsche nächste Nachbarn vorhanden sind.

Kulkarni et al. [Kulkarni97] schlägt vor, zur Abschätzung der Netzwerkarchitektur die Parameter der Einbettungsverzögerung und Attraktordimension zu verwenden. Für die zu untersuchenden Zeitreihen ergibt sich daraus, dass in der verdeckten Schicht vier bis sechs Neuronen ausreichen. In den Simulationen in Kapitel 3.2.4 stellt sich heraus, dass sich bereits mit zwei bis vier verdeckten Neuronen gute Ergebnisse erzielen lassen.

3.2.2 Kennzahlenextraktion mit Backpropagation

Zur Extraktion klassifikationsrelevanter Kennzahlen wird im Folgenden der Backpropagation-Algorithmus untersucht. Die Parameterkurven, die für das Training des neuronalen Netzes verwendet wurden, waren in zwei Klassen (*good*, *bad*) eingeteilt. Das Ziel ist, durch die Kennzahlenextraktion eine skalare Größe für die Güte der Prozessierung zu erhalten und dadurch die Klassifikation möglichst einfach zu ermöglichen. Nach Abschluss der Trainingsphase sollte das neuronale Netz in der Lage sein, auch untrainierte Parameterkurven zu bewerten.

Zur Festlegung der Architektur wurden die Ergebnisse der Datenvoranalyse verwendet. Eine zu geringe Zahl von Neuronen führt dabei dazu, dass das neuronale Netz nicht in der Lage ist die Trainingsdaten zu lernen. Eine zu hohe Zahl von Neuronen in den Schichten gehen auf Kosten der Rechenzeit und Generalisierungsleistung.

Zur Festlegung der Eingabeschicht wurde die Autokorrelation und damit die Redundanz innerhalb der Zeitreihen berücksichtigt. Es werden 100 Eingabeneuronen verwendet. Die Anzahl der verdeckten Neuronen ist abhängig von der Komplexität der Trainingsaufgabe. In unserem Fall müssen 2-6 Neuronen ausreichen. Da für die gestellte Aufgabe nur zwei Klassen separiert werden müssen, wird als Ausgabeschicht nur ein einzelnes Neuron verwendet.

Die freien Parameter, Lernrate η und Momentum-Term α, werden experimentell ermittelt. Für η werden Werte zwischen 0,1 und 0,2, für α Werte zwischen 0.8 und 0,9 verwendet.

Backpropagation konvergiert nur sehr langsam. Die verwendete *Online*-Lernregel verstärkt dies noch. Bei jedem Zyklus werden die einzelnen Gewichte immer wieder auf ein anderes Trainingsmuster eingestellt. Der Vorteil dieses Vorgehens gegenüber der *Offline*-Lernregel ist jedoch, dass lokale Minima der Fehlerfunktion leichter verlassen werden können.
Durch die Einführung eines Momentum-Term kann die Konvergenz erheblich beschleunigt werden [Botha98, Zell97]. Der Momentum-Term berücksichtigt die Gewichtsänderung Δw_{ij} im vorangehenden Zyklus und bewirkt eine Glättung der Fehlerfunktion. Verändert sich nun ein Gewicht w_{ij} stets in eine Richtung, so wird bspw. eine starke Veränderung in die entgegengesetzte Richtung gedämpft.
Die Gewichtsänderung berechnet sich mit dem Momentum-Faktor α wie folgt:

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t) = \eta \delta_j x_i + \alpha \Delta w_{ij}(t) \qquad (3.16)$$

Eine weitere sehr einfache Möglichkeit zur Beschleunigung der Konvergenz wird durch die Addition eines kleinen Bias Wertes γ zur Ableitung $f'(a)$ erreicht. Gebiete mit flachem Anstieg (Flat-Spot) können so schneller verlassen werden [Botha98].

$$f'_\gamma(a) = f'(a) + \gamma \qquad (3.17)$$

Die Konvergenz des neuronalen Netzwerkes kann mit dem Momentum-Term und der Flat-Spot Eliminierung erheblich beschleunigt werden. Nach 5000 bis 10000 Lernzyklen stellt sich bei einer Anzahl von 20 – 40 Trainingsmustern ein gut konvergiertes neuronales Netzwerk ein. Die hierzu benötigte Rechenzeit liegt bei einem normalen PC (200Mhz, 64MByte RAM) bei wenigen Minuten (2-4 min).

Ein neuronales Netzwerk sollte in der Lage sein, einerseits Neues zu Lernen, aber trotzdem bereits Gelerntes nicht wieder zu vergessen. Beide Punkte arbeiten jedoch gegeneinander. Bei Backpropagation zieht die Plastizität zu Gunsten der Stabilität den kürzeren. Die starre Architektur eines Backpropagation-Netzwerkes ermöglicht es nicht, nach Abschluss des Trainings noch Neuronen für neue Trainingsmuster zu erzeugen. Ist Backpropagation einmal mit einer gewissen Anzahl von Trainingsmuster konvergiert und auf diese Muster adaptiert, so können neu hinzukommende Muster die Gewichte nicht mehr ändern. Zu Begin eines Trainingslaufes sollen die Gewichte mit möglichst kleinen Werten belegt sein. Dies ist nach abgeschlossenem Training dann nicht mehr der Fall.
Als Resultat ausgeprägter Gewichte sind auch die Aktivierungen $\boldsymbol{a}$ der Neuronen hoch. Die Ableitung der Aktivierungsfunktion $f'(\boldsymbol{a})$ wird dadurch sehr klein und somit sind die Gewichtsänderungen $\Delta w_{ij} = \eta \cdot \delta_j \cdot x_i$ nur noch sehr gering.

$$\delta_j = (t_j - y_j) \cdot f'(a_j)$$
$$\delta_k = \sum_j w2_{ij} \delta_j \cdot f'(a_k)$$

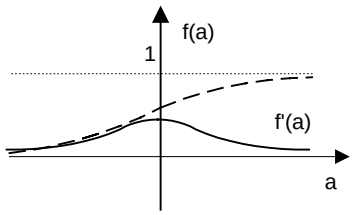

Abbildung 3.29: Sigmoide Aktivierungsfunktion mit Ableitung

Das Stabilitäts-Plastizitätsdilemma kann nicht gelöst werden. Sollen neue Trainingsmuster hinzugelernt werden, muss ein komplettes neues Training gestartet werden. Dies und die feste Architektur machen Backpropagation zu einem schlechten Klassifikator im Vergleich zu anderen Verfahren wie RBF oder RCE. Zur reinen Vorverarbeitung bzw. Kennzahlenextraktion der Daten eignet sich der Algorithmus jedoch sehr gut.

Für die Initialisierung der Gewichte werden üblicherweise kleine Zufallszahlen im Bereich von –1 bis +1 gewählt. Es zeigte sich, dass es auch möglich ist, die synaptischen Verbindungen der Eingabeschicht zur verdeckten Schicht mit 0 zu belegen. Durch diese Belegung kann die Ausprägung der Gewichte besser beobachtet werden und zur Analyse der Trainingsdaten (siehe nächstes Kapitel) herangezogen werden. Außerdem konvergierte das neuronale Netzwerk sogar geringfügig schneller. Für die Verbindungen der verdeckten Schicht zur Ausgabeschicht müssen in jedem Fall unterschiedliche Werte eingestellt werden.

Die Trainingsdaten für das neuronale Netzwerk sind zeitabhängige Kurven eines Parameters. Um den Zielwert für das Ausgabeneuron festzulegen (t=0 bzw. t=1) sind die Trainingskurven in 2 Klassen *good* und *bad* aufgeteilt. „*good*“-Parameterkurven werden mit t = 0, „*bad*“ mit t = 1 trainiert.

Folgendes Bild zeigt den gesamten Versuchsaufbau für die Kennzahlenextraktion mit Backpropagation für die *Emission_1*-Parameterkurve.

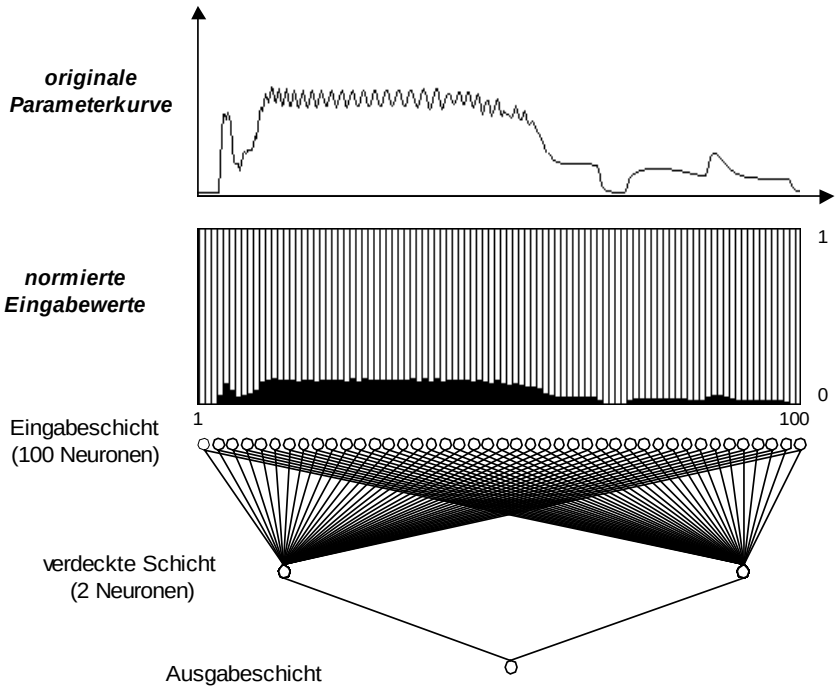

Abbildung 3.30: Versuchsaufbau für die Extraktion von Kennzahlen durch überwachtes Lernen bei Backpropagation-Algorithmen. Die Darstellung zeigt eine Originalparameterkurve, die normierten Eingabedaten als Eingaberaum des neuronalen Netzwerkes sowie die Netzwerkarchitektur mit einer verdeckten Schicht und einem Ausgabeneuron.

Pro Trainingszyklus wird abwechselnd ein *good*, und anschließend ein *bad* Datensatz zufällig dem neuronalen Netz präsentiert (*Online*-Lernverfahren). Dieses Vorgehen ermöglicht, dass die Anzahl der guten bzw. schlechten Trainingsmuster unerheblich ist. Dies ist insofern wichtig, da es (siehe Diskussion 3.1) wenige Prozessierungen mit geringer Ausbeute gibt. Würde das Verhältnis der Trainingsmuster der beiden Klassen nicht ausgewogen sein, so könnte dies dazu führen, dass eine Klasse bevorzugt trainiert wird und Repräsentanten der anderen Klasse nicht mehr erkannt werden.
Die Tests wurden mit 20 –100 Trainingsmustern durchgeführt. Im Normalfall konvergiert das neuronale Netz nach 5000 bis 10000 Lernzyklen hinreichend gut. Das sogenannte *Overfitting* konnte hierbei nicht beobachtet werden. Als Konvergenzkriterium wurde der quadratische Fehler über alle Trainingspaare herangezogen.

$$E = \sum_n (t_n - y_n)^2 \tag{3.18}$$

In der Recallphase wurde dem trainierten neuronalen Netz 197 unbekannte Kurven präsentiert. Das neuronale Netz reagiert auf jede Eingabe mit einem Ausgabewert im Bereich zwischen [0,1]. Je näher der Ausgabewert an 0 bzw. 1 liegt, desto sicherer ist die Zuordnung zu einer der Klassen *good* bzw. *bad*. Ein Wert zwischen 0 und 1 gibt an, dass das neuronale Netz nicht eindeutig in der Lage war die Kurve zuzuordnen. Die Ausgabe ist dann eine Interpretation der Kurve. Die Ausgabewerte des neuronalen Netzes liefert also keine scharfe Einordnung der Kurven in 2 Klassen, sondern gibt im weitesten Sinne eine *posterior* Wahrscheinlichkeit der Klassenzugehörigkeit an.

Abbildung 3.31 zeigt einen Scatterplot der Ausgabewerte des neuronalen Netzwerkes für 197 Testdaten des *Emission_1*-Parameters. Die Punkte, die nahe dem Wert 1 sind, wurden vom Netz als Ausreißer erkannt.

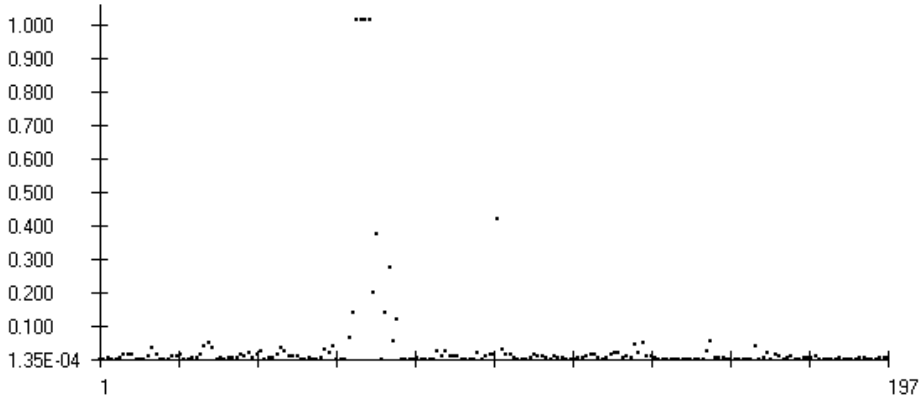

Abbildung 3.31: Ausgabewerte [0,1] des neuronalen Netzwerkes für 197 Testdaten des Emission_1-Parameters. Die Ausgabe nahe Null zeigt gute Prozessierungen, nahe Eins schlechte Prozessierungen an. Bei einer Ausgabe im Bereich von 0,5 kann das neuronale Netz keine Zuordnung treffen.

Die Zuordnung zu den beiden Klassen ist gut sichtbar. Bei Ausgabewerten zwischen 0,0 – 0,1 sind die Kurven glatt. Im Bereich von 0,1 – 0,5 nehmen die Oszillationen der Kurven im Bereich des *Overetch* zu und weisen auf Aluminiumreste auf den Scheiben hin. Der Ätzvorgang war also nicht erfolgreich. Die Oszillationen waren das Unterscheidungskriterium zwischen *good* und *bad*, wobei starke Oszillationen eine *bad*-Kurve charakterisieren.
Im Bereich um 1,0 schließlich sind die stark oszillierenden Kurven deutlich als Ausreißer erkennbar.

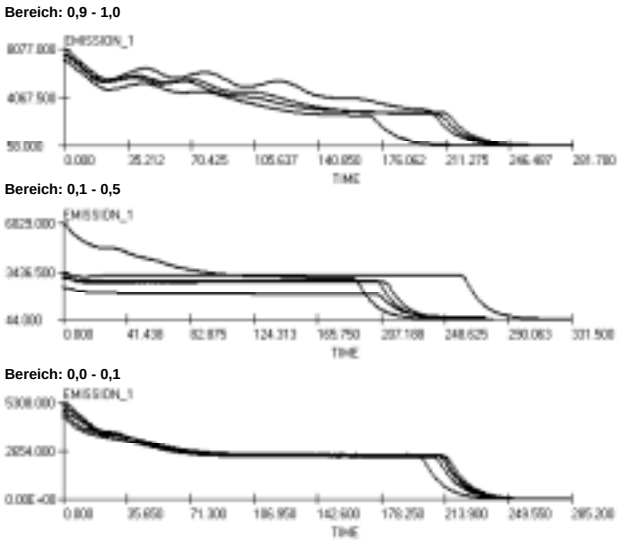

Abbildung 3.32: Original Zeitreihen des Overetch im Emission_1-Signals und der Ausgabewert des neuronalen Netzwerkes. Parameterverläufe im Bereich zwischen 0,9 und 1,0 weisen starke Oszillationen auf, die bei dieser Ätzung auf Aluminiumreste auf den Scheiben hinweisen. Gezeigt ist der Overetch des Emission_1-Signals.

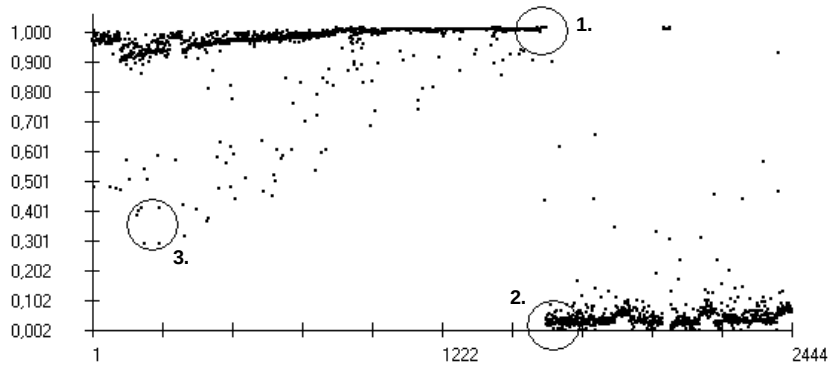

Abbildung 3.33: Ausgabewerte [0,1] des neuronalen Netzwerkes für 2444 Testdaten des RF_Match_Tuning_Position Parameters. Auch hier zeigt die Ausgabe nahe Null gute Prozessierungen, nahe Eins schlechte Prozessierungen an. Bei einer Ausgabe im Bereich von 0,5 kann das neuronale Netz keine Zuordnung treffen.

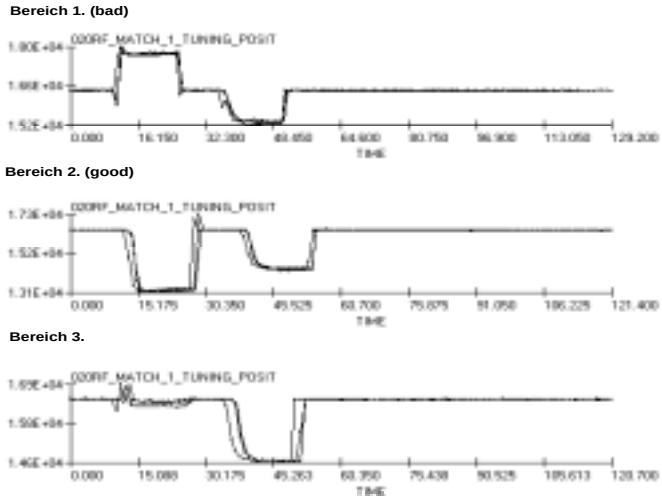

Abbildung 3.34: Original Zeitreihen und der Bereich der Ausgabe des neuronalen Netzwerkes. Kurven im Bereich 1 stellen schlechte, im Bereich 2 gute Parameterverläufe dar. Die Signalverläufe im Bereich 3 wurden dem neuronalen Netzwerk nicht trainiert, was zu einem Ausgabewert um 0,5 führt.

Die Extraktion von Kennzahlen mit Backpropagation-Algorithmus wurde für einen weiteren Parameter, *RF_Match_Tuning_Position*, durchgeführt. 40 Kurven aus Bereich 1 in Abbildung 3.33, wurden als Trainingsdaten für die Klasse *bad* verwendet, 40 Kurven aus Bereich 2 für *good*. Im Bereich 3 sind Kurven die nicht zugeordnet werden konnten. Bei genauerer Betrachtung der Kurven aus Bereich 3 wird offensichtlich, dass sie sich erheblich von den trainierten Kurven aus Bereich 1 und 2 unterscheiden und das neuronale Netz deshalb nicht in der Lage sein kann, eine eindeutigere Zuordnung durchzuführen.

Dieses Ergebnis ist insofern interessant, als das Netzwerk in der Lage ist, bekannte Abnormitäten bzw. normale Zustände zu klassifizieren. Unbekannte Kurvenverläufe werden tatsächlich auch als unbekannt im Sinne einer Netzwerkausgabe von 0,5 indiziert. Durch dieses Vorgehen erreicht man nicht nur eine Einteilung in zwei Klassen, sondern erhält ein Ausgabekontinuum, das für eine eindeutige Eingabe eine eindeutige Zuordnung und für eine

unsichere Eingabe die Wahrscheinlichkeit einer eher guten bzw. eher schlechten Prozessierung angibt.

3.2.3 Verfahren zur Parameterkurven-Analyse mit Backpropagation

Beim Training der Parameterkurven werden die synaptischen Verbindungen so eingestellt, dass das neuronale Netz sich den Trainingskurven anpasst. Die anfänglich mit kleinen Zufallszahlen oder mit Null initialisierten Gewichtswerte in den Gewichtsmatrizen, werden teilweise stark positiv oder negativ. Erste graphische Darstellungen der Gewichtswerte zeigen die Ausprägung der Gewichte zu Mustern. Interessant sind im Folgenden die Synapsen von der Eingabeschicht zu den einzelnen Neuronen der verdeckten Schicht.
In Abbildung 3.35 ist eine farblich codierte Darstellung der Gewichtswerte in einer Gewichtsmatrix für 100 Eingabeneuronen und 2 verdeckte Neuronen eines trainierten neuronalen Netzwerkes zu sehen.

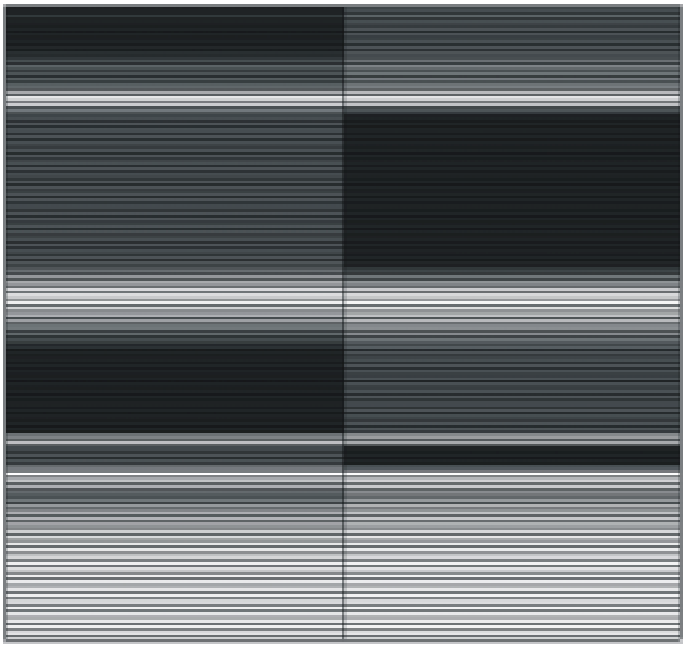

Abbildung 3.35: Farblich codierte Gewichtswerte einer 2x100 Gewichtsmatrix. Schwarz entspricht negativen und weiß positiven Gewichtswerten. (Gewichte von der Eingabe- zur verdeckten Schicht)

Bei einer Darstellung als Balkendiagramm in Abbildung 3.36 wird die auffällige Ausprägung der Synapsen schließlich noch deutlicher erkennbar. Die Beobachtungen der Gewichte zeigen, dass besonders bei den Verbindungen zu Eingabeneuronen, an denen sich die Eingabewerte der Kurven der Klassen *good* und *bad* unterscheiden, eine starke synaptische Ausprägung entsteht.
Dies sind schließlich auch die Kurvenbereiche, welche zur Unterscheidung der Kurven herangezogen werden.

Die Gewichte w_i eines Neurons c_i der verdeckten Schicht werden beim Backpropagation-Algorithmus nach folgender Formel adaptiert:

$$w_i(t+1) = w_i(t) + \delta_{2j} \cdot \eta \cdot x_i \qquad (3.19)$$

wobei $\boldsymbol{x}$ eine normierte Trainingskurve ist.

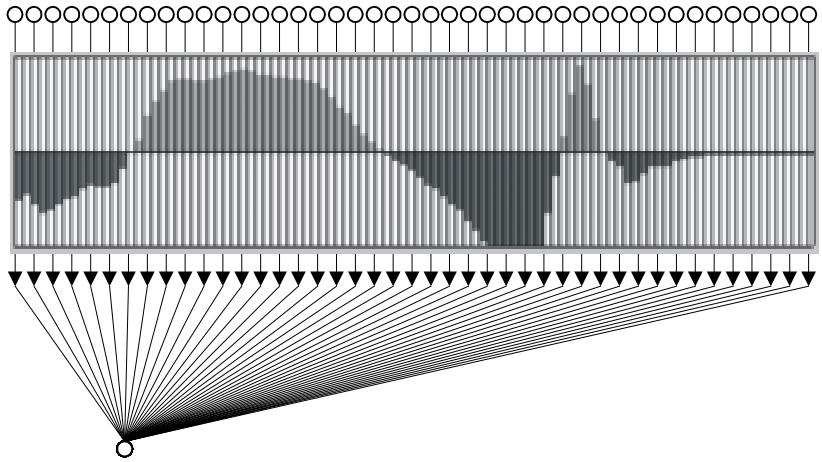

Abbildung 3.36: Gewichtsvektor als Balkendiagramm dargestellt.

Der Fehler δ_{2j}, den das Neuron c_j gemacht hat, wird aus dem Fehler δ_3, der durch den Eingabevektor $\boldsymbol{x}$ am Ausgabeneuron verursacht wird, berechnet.

$$\delta_{2j} = w2_j \cdot \delta_3 \cdot f'(a_j) \qquad (3.20)$$

δ_3 ist der Fehler an der Ausgabeschicht bzw. die Differenz der Sollausgabe t und der Istausgabe y des neuronale Netzwerkes als Reaktion auf die Eingabekurve $\boldsymbol{x}$.

$$\delta_3 = (t-y) \cdot f'(y) \qquad (3.21)$$

Die verwendeten neuronalen Netzwerk-Architekturen haben nur ein Ausgabeneuron. Somit geschieht die Einteilung der Eingabekurven in 2 Klassen durch die Zielwerte $t = 0$ und $t = 1$.
Es gibt also n Eingabekurven $\boldsymbol{x}_{0n}$, die mit Zielwert $t=0$ und n Eingabekurven $\boldsymbol{x}_{1n}$, die mit Zielwert $t=1$ trainiert werden.

Mit einer Sigmoid-Funktion als Ausgabefunktion können die Ausgabewerte eines Neurons nur positive Werte zwischen 0 und 1 annehmen. Da auch die Ableitung $f'(a)$ der Ausgabefunktion immer positiv ist, lässt sich folgende einfache Aussage über das Vorzeichen von δ_3 machen.

mit

$$0 \le y \le 1$$
$$f'(a) \ge 0$$

gilt

Fall1:

$$t = 0$$
$$\Rightarrow \delta_3 = (t - y) \cdot f'(y) \leq 0 \tag{3.22}$$

Fall2:

$$t = 1$$
$$\Rightarrow \delta_3 = (t - y) \cdot f'(y) \geq 0 \tag{3.23}$$

Für Eingabekurven $\boldsymbol{x}_0$ der Klasse *good* ist der Fehler der Ausgabeschicht δ_3 somit immer negativ und für Eingabekurven $\boldsymbol{x}_1$ der Klasse *bad* positiv.
Für ein Neuron der verdeckten Schicht, welches über die synaptische Verbindung $w2_j$ mit dem Ausgabeneuron verbunden ist, wird diese Beziehung für Eingabekurven und δ_3 somit auf δ_{2j} übertragen.

$$\delta_{2j} = w2_j \cdot \delta_3 \cdot f'(a_j) \tag{3.24}$$

Ist nun $w2_j$ positiv, so gilt für die Berechnung der synaptischen Verbindungen des Neurons c_j der verdeckten Schicht zur Eingabeschicht folgendes (analog für $w2_j$ negativ):

Für $\boldsymbol{x_0} \in X0$, mit *X0* der Menge aller guten Prozessierungen, gilt:
Fall1:

$$t = 0$$
$$\delta_3 \leq 0 \Rightarrow \delta_{2j} \leq 0 \tag{3.25}$$
$$\Rightarrow \mathbf{w}_j(t+1) = \mathbf{w}_j(t) - \left|\delta_{2j}\right| \cdot \eta \cdot \mathbf{x}_0$$

Für $\boldsymbol{x_1} \in X1$, mit *X1* der Menge aller abnormalen Prozessierungen, gilt:
Fall2:

$$t = 1$$
$$\delta_3 \geq 0 \Rightarrow \delta_{2j} \geq 0 \tag{3.26}$$
$$\Rightarrow \mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \left|\delta_{2j}\right| \cdot \eta \cdot \mathbf{x}_1$$

Da zu Beginn des Trainings die Gewichte $\boldsymbol{w}$ mit kleinen Werten bzw. mit Null belegt sind und dann in vielen Iterationsschritten abwechselnd Trainingspaare $\{\boldsymbol{x}_0, t=0\}$ und $\{\boldsymbol{x}_1, t=1\}$ dem neuronalen Netz präsentiert werden, repräsentieren die Gewichte von der Eingabeschicht zur verdeckten Schicht dann, nach Abschluss des Trainings, die Differenz der *n* Trainings-Eingabekurven $\boldsymbol{x}_0$ und $\boldsymbol{x}_1$.

Die Gewichte spiegeln dabei nicht die exakte Differenz der *n* Trainings-Eingabekurven $\boldsymbol{x}_0$ und $\boldsymbol{x}_1$ wider. Jede Eingabekurve erzeugt am Ausgabeneuron einen bestimmten Ausgabewert *y*. Somit ist δ_{2j} abhängig von der jeweiligen Eingabekurve. Zusätzlich haben auch noch die

anderen Neuronen der verdeckten Schicht (die nicht betrachtet wurden) einen Einfluss auf δ_{2j}. Die Betrachtung der Vorzeichen von δ_3 und δ_{2j} gelten aber in jedem Fall.

Im folgenden Versuch wurde dem neuronalen Netz eine Sinuskurve als $\boldsymbol{x}_1$ und eine wagrechte Gerade als $\boldsymbol{x}_0$ trainiert. Nach wenigen Lernzyklen stellen sich die Gewichte, wie erwartet, ein.

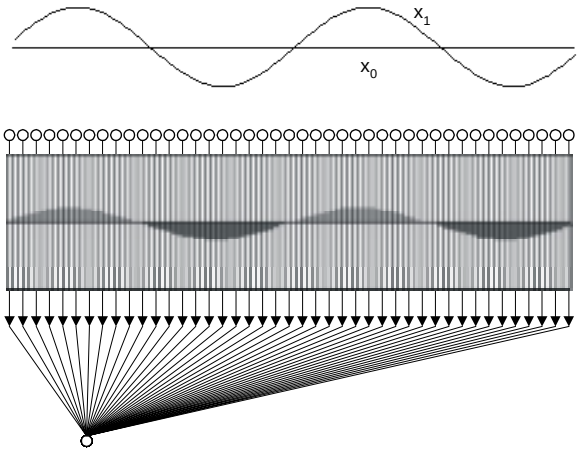

Abbildung 3.37: Trainingsmuster ($\boldsymbol{x}_0$, $\boldsymbol{x}_1$) und dazu ausgeprägte synaptische Verbindungen.

Die Eigenschaften der Trainingskurven spiegeln sich in den Gewichten von der Eingabeschicht zur verdeckten Schicht wider. An den Stellen, wo sich die Kurven der beiden Trainingsklassen unterscheiden, werden sich auch die Verbindungen der zuständigen Neuronen stärker ausprägen. Stellen, an denen keine signifikanten Unterschiede zwischen den Kurven existieren, sind nur schwach ausgeprägt, d.h. signifikante Abschnitte der Kurven können ermittelt werden. Oder anders formuliert heißt das, dass für Parameterkurven, deren Synapsen in den neuronalen Netzwerken schwach ausgeprägt sind, nicht für die Unterscheidung zwischen *good* bzw. *bad* herangezogen werden kann.

Die Parameterkurven sind in mehrere Prozessierungsschritte (sogenannte *Steps*) unterteilt. Im Folgenden soll untersucht werden, ob sich die synaptischen Gewichte zur Detektion des relevanten *Steps* zur Klasseneinteilung eignen.
Zur Analyse der Gewichte wird eine einfache Aufsummierung aller Gewichtsvektoren von der Eingabeschicht zur verdeckten Schicht durchgeführt. Daraus ergibt sich ein Vektor MSUM. In den einzelnen Abschnitten der Kurven wird nun der Mittelwert der entsprechenden Komponenten von MSUM gebildet. Dieser Wert pro Abschnitt (*Step*) gibt an, wie groß die synaptische Ausprägung an dieser Stelle ist.

$$MSUM_i = \sum_j \left| w_{ij} \right| \tag{3.27}$$

Die folgende Abbildung 3.38 zeigt eine Analyse des *Emission_1*-Parameters. Auffälligster *Step* ist eindeutig Nummer 4.

Bei jeder Prozessierung werden mehrere Parameterkurven aufgezeichnet und in einem Rohdatenfile abgespeichert. Nach der Idee der Synapsen-Analyse wird ein neuronales Netzwerk, welches auf einen Parameter trainiert ist, der keine aussagekräftigen Daten zur Unterscheidung in 2 Klassen *good* und *bad* enthält, nur schwach ausgeprägte synaptische Gewichte aufweisen.
Für jeden in den Rohdaten aufgezeichneten Parameter wird nun ein eigenes neuronales Netz trainiert. Anschließend führt eine Synapsen-Analyse bei den einzelnen Netzwerken zu einem Wert für die einzelnen synaptischen Ausprägungen. Die Parameter können nach diesen Werten geordnet werden. Parameter, deren Netzwerke gut ausgeprägte Synapsen aufweisen, können besser zur Unterscheidung der Prozessierung normal bzw. abnormal herangezogen werden.

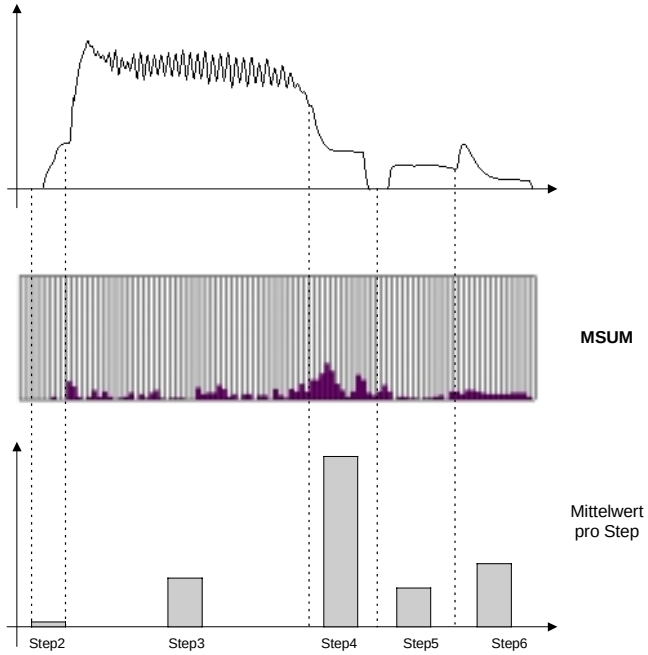

Abbildung 3.38: Summenvektor MSUM aller Gewichtsvektoren von Eingabe- zur verdeckten Schicht und Mittelwertbildung der Vektorkomponenten pro Kurvenabschnitt (Step).

Eine weitere Möglichkeit zur Beurteilung der Trainingsdaten ist der Fehlerverlauf beim Training des neuronalen Netzes. Der Fehler am Ende des Trainings gibt an, wie gut das neuronale Netz in der Lage war, sich an die Trainingsdaten zu adaptieren. Gibt es keine Unterschiede der Kurven in den Klassen *good* bzw. *bad*, so kann diese Adaption nur schlecht stattfinden. Somit enthalten solche Parameterkurven keine Information, um zwischen normaler und abnomaler Prozessierung zu unterscheiden.

Diese beiden Kriterien zusammen betrachtet ermöglichen es, Aussagen über die aufgezeichneten Daten zu machen. Eine Maßzahl für die Signifikanz eines Parameters kann folgendermaßen definiert werden:

$$P = a \cdot SynA + b \cdot (1 - Sqerr) \qquad (3.28)$$

Hierbei gibt *SynA* den Mittelwert des signifikanten *Steps* eines Parameters an. *Sqerr* ist der Quadratische Fehler des Netzwerkes für den Parameter. a und b sind Gewichtungen die angeben, wie stark die Synapsen-Analyse oder die Konvergenz des neuronalen Netzes in das Ergebnis einfließen soll.
Für die vorhandenen Testdaten von zwei unterschiedlichen Anlagen wurde diese Analyse durchgeführt. Das Ergebnis ist in Tabelle 3.10 für die Untersuchung an Anlage 1 und Separation aufgrund des *RF_Match_Tuning_Position*-Parameters bzw. Tabelle 3.11 für Anlage 2 und der Klasseneinteilung aufgrund des *Emission_1*-Parameters (siehe auch Abbildung 3.32 und 3.34) ersichtlich.

Tabelle 3.11: Analyseergebnis zur Auswertung signifikanter Parameter bei Anlage 1 50 Runs, 5000 Lernzyklen, a=0,7, b=1,4

Parameter	Step	SynA	Sqerr	P
RF_MATCH_1_TUNING_POSIT	3	6.418	0.029672037	0.1937
HE	7	3.659	0.082328971	0.1687
PRES_CNTRL_POSITION	4	4.818	0.272822949	0.1452
AR	7	3.311	0.181832133	0.1507
CHAMBER_PRESS_AI	4	2.524	0.215208704	0.1407
RF_FORWARD_POWER_AI	3	1.891	0.21481343	0.1369
CHAMBER_PRESSURE_MT	4	2.37	0.23387338	0.1368
RF_REVERSE_POWER_AI	5	1.588	0.20528544	0.1366
CF4	7	2.042	0.241567247	0.1336
CLAMP_FLOW_AI	7	1.472	0.245557101	0.1295
CF4	2	1.368	0.248806671	0.1283
RF_MATCH_1_LOAD_COIL_PO	3	0.869	0.24816967	0.1255
O2	8	0.619	0.243632137	0.1247
CHF3	3	0.604	0.251052116	0.1234
ELL_VAC_SENS_AI	8	0.519	0.248813416	0.1233
WAFER_NUMBER	8	0.283	0.252438473	0.1213
NF3	1	0	0.252583288	0.1195
ENDPT_DETECT_A_AI	1	0	0.252583288	0.1195
ENDPT_DETECT_C_AI	1	0	0.252583288	0.1195
ENDPT_DETECT_D_AI	1	0	0.252583411	0.1195
RF_GEN_2_REFL_AI	1	0	0.252583288	0.1195
RF_GEN_2_FWD_AI	1	0	0.252583288	0.1195
ISO_PLASMA_DETECTOR_AO	1	0	0.252583288	0.1195

Der führendende Parameter *RF_Match_Tuning_Position* ist derjenige, bei dem die Unterscheidung am deutlichsten erkennbar ist. Die *Pres_Ctrl_Position* und der Heliumfluss (He) sind für die Separation auch noch geeignet und korrelieren mit der Klasseneinteilung. Dieses Ergebnis stimmt auch mit den Beobachtungen der Prozesstechnik überein.

Durch die manuelle Auswertung eines Prozesstechnikers für Anlage 2 vermutete man als den relevantesten Parameter für „Problemlose“ die Oszillationen am *Overetch* des *Emission_1*-Signals. Durch die automatische Auswertung wird deutlich, dass tatsächlich beim *Step* 4, und damit beim *Overetch*, die Separation der Klassen am deutlichsten zu sehen ist. Interessant ist allerdings, dass der Parameter *ION_.001MT* auch eine relativ große Korrelation mit der Klasseneinteilung besitzt (siehe Tabelle 3.12), die Abnormität allerdings bereits im ersten *Step* auffällig ist. Diese Information ist insofern äußerst interessant, da hier evtl. interne Abhängigkeiten vorhanden sind, die man durch eine manuelle Bewertung der Kurvenverläufe nicht bemerkt. Darüber hinaus ist die Auffälligkeit des Wafers bereits zu Beginn der Prozessierung bei *Step* 1 bemerkbar und somit könnte der Wafer evtl. vor der drohenden Fehlprozessierung gerettet werden.

Tabelle 3.12: Analyseergebnis zur Auswertung signifikanter Parameter bei Anlage 2 50 Runs, 1000 Lernzyklen, a=0.6, b=1.6

Parameter	Step	SynA	Sqerr	P
BCL3	4	10.96	0.129793124	0.1985
EMISSION_1	4	7.92	0.030564806	0.1911
ION_.001MT	1	9.095	0.145061178	0.1833
THROTTLE_VALVE	4	6.622	0.104300968	0.1717
RF_REFLECTED	6	6.048	0.07692489	0.1715
CHF3	4	7.645	0.173115645	0.1692
PROCCHAMPRESS	4	6.202	0.113496726	0.1675
CL2	4	7.803	0.272463811	0.1564
PRESSURE_MANO	4	5.472	0.186472061	0.1521
O2	4	5.123	0.191563889	0.149
DC_BIAS	4	7.156	0.308511792	0.1469
FORELINE_PR	1	5.976	0.253465185	0.1463
CF4	4	6.259	0.284194803	0.144
HEATEXCHTEMP	3	3.901	0.175066145	0.1427
EMISSION_2	1	4.862	0.31219065	0.1303
LOADCHAMPRESS	2	2.233	0.241979957	0.1217
ROUGH_MT	6	2.463	0.263581795	0.1203
GATEVALVETEMP	1	1.372	0.255587014	0.1138
CRYO_TEMP_K	6	1.257	0.256401448	0.1129
MANO_OFFSET_MT	1	1.269	0.256473948	0.1129
TURBO_PURGE_CC	3	1.271	0.256476707	0.1129
CRYO_MT	1	0	0.257438821	0.1039
N2_HAND	1	0	0.257435041	0.1039

3.2.4 Datenanalyse mit selbstorganisierenden Algorithmen

Zur Analyse zeitabhängiger Parameterkurven werden selbstorganisierende Verfahren untersucht, die auf Kohonennetzen basieren. Auch bei diesen Untersuchungen besteht die Vorverarbeitung aus der Betragsnormierung und Diskretisierung der Parameterkurven, wie bereits im Abschnitt über Backpropagation gezeigt. Ziel des Verfahrens ist, Cluster von zusammengehörenden (ähnlichen) Kurven zu erhalten. Die resultierenden Informationen aus diesem Verfahren können dann z. B. für das Training der Backpropagation-Netze eingesetzt werden.

Das neuronale Netzwerkmodell von Toivo Kohonen orientiert sich besonders stark am biologischen Vorbild. Bei Lebewesen ist bekannt, dass sich für die taktilen Sinne der Haut und der Organe auf entsprechende Neuronen der Großhirnrinde wiederfinden. Hierbei spiegelt die Position der Neuronen auf der Großhirnrinde (dem Kortex) die Position der Sinneszellen auf der Haut wider. Die Haut und Organe werden somit auf den Kortex abgebildet.

Diese Art der topologieerhaltenden Abbildung wird auch von Kohonenkarten realisiert. Ein n-dimensionaler Eingaberaum wird hierbei topologieerhaltend auf die Kohonenkarte abgebildet. Das heißt, dass sich benachbarte Bereiche im n-dimensionalen Eingaberaum auf der m-dimensionalen Merkmalskarte ebenfalls in benachbarten Gebieten befinden werden. Ist die Dimension der Ausgabeschicht kleiner als die Dimension der Eingabeschicht, kann dieses Verfahren auch zur Dimensionsreduzierung verwendet werden und dadurch z. B. die Visualisierung höherdimensionaler Räume unterstützen.
In den folgenden Versuchen wurde nur mit zweidimensionalen Kohonenkarten gearbeitet. Das Netzwerk besteht dann aus einer n-dimensionalen Eingabeschicht und einer

zweidimensionalen Ausgabeschicht (siehe Abbildung 3.39). Jedes Ausgabeneuron der Kohonenkarte ist über seine Synapsen mit der Eingabeschicht verbunden. Zusätzlich ist jedes Ausgabeneuron durch laterale, bzw. *intralayer* Verbindungen, mit allen anderen Ausgabeneuronen verbunden. Diese Verbindung wird durch die Nachbarschaftsfunktion *h* realisiert.

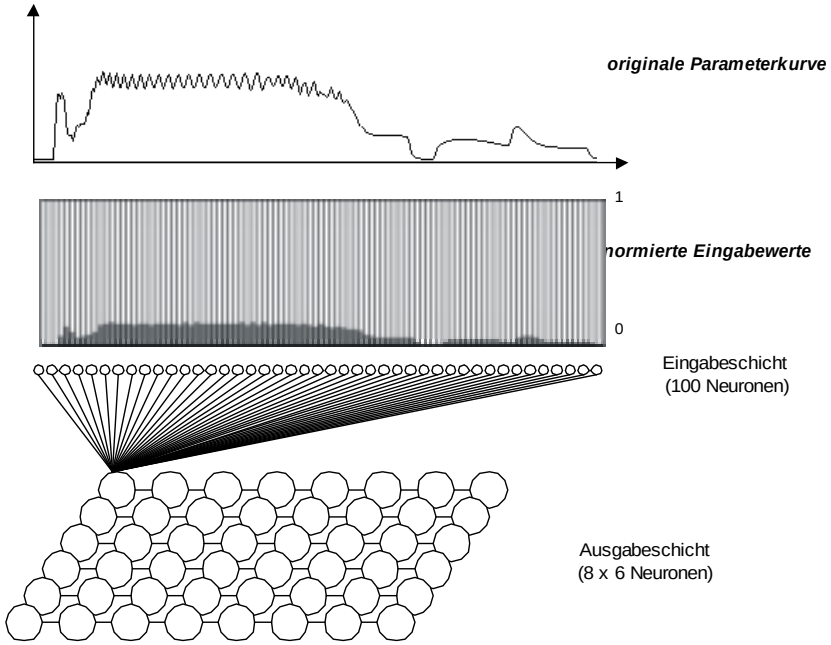

Abbildung 3.39: Architektur der verwendeten Kohonenkarte.

Zur Initialisierung der Gewichtsvektoren werden kleine positive Zufallszahlen gewählt. Die Aktivierungen der Neuronen der Kohonenkarte werden mittels der euklidischen Distanz des Eingabevektors zu den Gewichtsvektoren ermittelt. Das Lernen lässt sich dabei in zwei Schritte pro Lernzyklus unterteilen.

Die Aktivierung Φ_i jedes Neurons der Kohonenkarte wird mit der euklidischen Distanz zwischen Gewichts- bzw. Referenzvektor $\boldsymbol{q}_i$ und Eingabevektor $\boldsymbol{x}$ ermittelt.

$$\Phi_j = \sqrt{\sum_i \left(x_i - q_{ij}\right)^2} \tag{3.29}$$

Sieger wird das Neuron der Kohonenkarte mit der kleinsten Aktivierung. Der Gewichtsvektor des Siegerneurons hat also den kleinsten euklidischen Abstand zum Eingabevektor und ist diesem somit am ähnlichsten.

Nach der Ermittlung des Siegerneurons werden die Gewichte der Neuronen adaptiert. Das Siegerneuron *s* fungiert hierbei als Erregungszentrum *Z* der Kohonenkarte. Je näher ein Neuron der Kohonenkarte diesem Erregungszentrum ist, desto stärker wird sein Referenzvektor verändert.

Als Nachbarschaftsfunktion *h* wurde die Gauß'sche Glockenkurve gewählt. Bei Versuchen mit der Mexican-Hat Funktion wurden keine verwertbaren Ergebnisse erzielt.

$$h_{js}(t) = e^{-\left(\frac{d_{js}}{\sigma(t)}\right)^2} \tag{3.30}$$

mit

d_{js}: Abstand des Neurons c_j zum Erregungszentrum Z
$\sigma(t)$: zeitlich variabler Einflussbereich des Erregungszentrums Z

In Abbildung 3.40 ist eine Kohonenkarte mit einem Erregungszentrum Z und dem Einfluss, über die Nachbarschaftsfunktion *h*, auf benachbarte Neuronen dargestellt.
Visualisiert ist nur ein diagonaler Schnitt durch die 3-dimensionale Gauß'sche Glockenkurve.

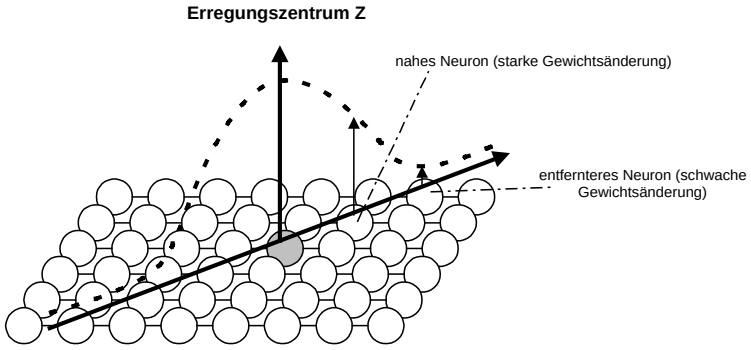

Abbildung 3.40: Nachbarschaftsfunktion h mit Siegerneuron als Erregungszentrum.

Die Gewichtsänderung bewirkt, dass der Gewichtsvektor jedes Neurons sich in Richtung des Eingabevektors bewegt. Dem Erregungszentrum nahe Neuronen werden hierbei ihre Gewichtsvektoren stärker verändern als entfernte Neuronen. Mit zunehmendem Lernen werden die Einflüsse der Erregungszentren auf benachbarte Neuronen immer schwächer. Der Referenzvektor $\boldsymbol{q}_j$ des Neurons c_j wird nach folgender Formel adaptiert:

$$\mathbf{q}_j(t+1) = \mathbf{q}_j(t) + \eta(t) \cdot h_j(t) \cdot \left(\mathbf{x} - \mathbf{q}_j(t)\right) \tag{3.31}$$

Das Ende des Lernvorgangs ist erreicht, falls eine maximale Anzahl von Zyklen erreicht ist, oder das Netzwerk vollständig konvergiert ist.
Durch die Modifikation der Gewichtsvektoren in Richtung der Eingabevektoren wird erreicht, dass sich nach Abschluss des Lernens in den Referenzvektoren $\boldsymbol{q}_j$ den Eingabekurven ähnliche Muster ausprägen. Dies hat natürlich den Vorteil einer einfachen Visualisierung des Ergebnisses der Kohonenkarte (siehe Abbildung 3.41).
Für die Lernrate η wurde in den Versuchen in diesem Abschnitt mit folgenden stetig fallenden Funktionen experimentiert.

-Gerade:

$$\eta(t) = \eta_0 \left(1 - \frac{step}{Cyclen} \right) \tag{3.32}$$

-Parabel:

$$\eta(t) = \eta_0 \left(1 - \left(\frac{step}{Cyclen} \right)^2 \right) \tag{3.33}$$

Die Parabel ist für das Verfahren günstiger, da die Größe der Gewichtsadaption nicht so schnell gegen Null geht und der Kohonenkarte mehr Zeit bleibt, sich vollständig auszuprägen.

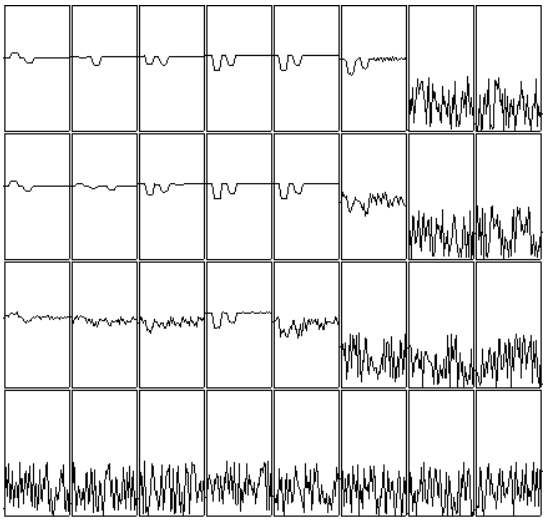

Abbildung 3.41: Gewichtsvektoren einer teilweise ausgebildeten Kohonenkarte. Diese Darstellung der Gewichtsmatrix ermöglicht einen einfachen Überblick über die unterschiedlichen Klassen der Trainingsdaten. Bereits in diesem frühen Stadium sind die unterschiedlichen Klassen deutlich zu erkennen (oben links, oben mitte).

Eine zu große Anzahl von Ausgabeneuronen in der Kohonenkarte bewirkt, dass einzelne Eingabemuster direkt auf Ausgabeneuronen abgebildet werden. Dies bedeutet, dass die Generalisierungsleistung des neuronalen Netzes nachlässt. Das Problem ist aber nicht so gravierend, da das Netzwerk in diesem Beispiel nicht zum Klassifizieren von Kurven, sondern primär zur Analyse der Menge von Eingabekurven benutzt wird. Wegen des steigenden Rechenaufwands bei vielen Neuronen ist es unnötig, mehr Ausgabeneuronen als Eingabemuster zu verwenden.

In Abbildung 3.41 sind die Gewichtsvektoren einer teilweise ausgebildeten Kohonenkarte dargestellt. Diese Darstellung der Gewichtsmatrix ermöglicht einen einfachen Überblick über die unterschiedlichen Klassen der Trainingsdaten. Bereits in diesem frühen Stadium sind die unterschiedlichen Klassen deutlich zu erkennen und es zeigt sich, dass die Kohonenkarte in der Lage ist, die unterschiedlichen Klassen der Signalverläufe zu separieren.

Bei der Wahl der freien Parameter des Netzes (Anzahl der Neuronen der Kohonenkarte, Anzahl der Zyklen, η_0) sollte man allerdings darauf achten, dass sich die SOM vollständig

ausprägen kann. Dies bedeutet, dass sich alle Gewichte auf Eingabekurven eingestellt haben. Sind die freien Parameter falsch gewählt, wird sich die SOM nicht vollständig ausbilden können (Abbildung 3.41).

3.2.5 Selbstorganisierende Oberflächen

Bereits zu Beginn der Implementierung der Kohonenkarte war die Idee entstanden, nicht nur die Gewichtsvektoren dem Eingabemuster anzunähern, sondern auch die Position der Neuronen auf der SOM in Richtung auf das Siegerneuron zu modifizieren.
Dies bewirkt eine Annäherung benachbarter Neuronen und somit einen sich verstärkenden Einfluss der Nachbarschaftsfunktion. Die feste Position im Neuronengitter wie bei der Kohonenkarte geht allerdings verloren. Die Neuronen sind auf der Oberfläche der ehemaligen SOM frei beweglich wie Teilchen auf einer Flüssigkeitsoberfläche.

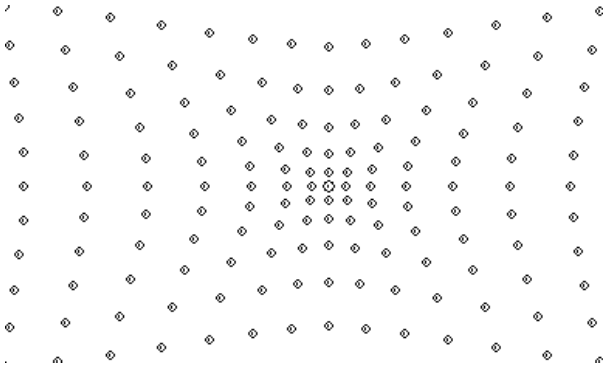

Abbildung 3.42: Sich auf das Siegerneuron zu bewegende Neuronen einer SOS.

Zell [Zell97] beschreibt ein diesem Vorgehen prinzipiell ähnliches Verfahren der Self Organizing Surface (SOS). Abbildung 3.43 zeigt den ersten Zyklus eines Lernvorgangs. Das Siegerneuron zieht gemäß der Gauß'schen Nachbarschaftsfunktion seine Nachbarn an. Zusätzlich werden auch deren Gewichtsvektoren mit dem Kohonen-Algorithmus verändert.

Jedes Neuron braucht nun zusätzlich 2 Koordinaten *(x,y)*, die seine Position auf der flexiblen Oberfläche festlegt. Die Nachbarschaftsfunktion *h* kann analog zu Kohonen berechnet werden. Die Distanz d_{js} zwischen Siegerneuron und anderen Neuronen ist einfach die euklidische Distanz auf der Oberfläche:

$$h_j(t) = e^{-\left(\frac{d_{js}}{\sigma(t)}\right)^2} \tag{3.34}$$

Die Gewichtsänderung wird ebenfalls analog zur Kohonenkarte berechnet:

$$\mathbf{q}_j(t+1) = \mathbf{q}_j(t) + \eta(t) \cdot h_j(t) \cdot \left(\mathbf{x} - \mathbf{q}_j(t)\right) \tag{3.35}$$

Einziger Unterschied zum SOM-Algorithmus ist, dass sich nun pro Zyklus auch die Position (*x*,*y*) der Neuronen auf der Oberfläche verändert:

$$x_j(t+1) = x_j(t) + v \cdot \eta(t) \cdot h_j(t) \cdot \Delta x_{jZ} \tag{3.36}$$

$$y_j(t+1) = y_j(t) + v \cdot \eta(t) \cdot h_j(t) \cdot \Delta y_{jZ} \tag{3.37}$$

Δx_{jZ} und Δy_{jZ} geben die x- und y-Richtung an, in die das Neuron bewegt werden muss. Wegen der zyklischen Randbedingungen sind diese nicht einfach nur die Differenz zwischen den *x*- und *y*-Koordinaten des Siegerneurons und der anderen Neuronen. Der Faktor *v* (Viskosität) gibt an, wie stark die Beweglichkeit der Neuronen auf der Oberfläche ist.

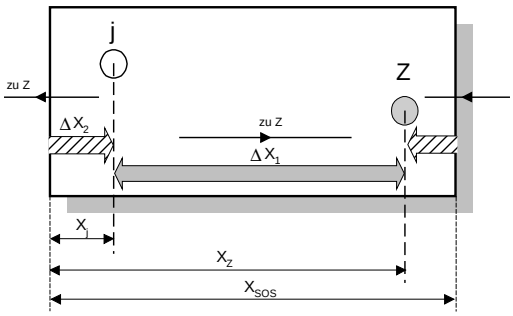

Abbildung 3.43: Distanzberechnung zwischen zwei Punkten auf der geschlossenen SOS. Δx_1 *ist die direkte Distanz,* Δx_2 *ist die Distanz über den Rand.* x_{sos} *ist die Kantenlänge der Oberfläche in x-Richtung.*

Neuronen, die an den Rändern der Oberfläche liegen, haben nach außen weniger oder keine Nachbarn. Demzufolge würden sie sich also immer nur nach innen zum Zentrum der Oberfläche hinbewegen. Dies führt wiederum zu einer noch größeren Dichte von Neuronen im Zentrum, und somit zu einer stärkeren Anziehung aus diesem Gebiet auf außenliegende Neuronen.
Resultat dieser Anziehung des Zentrums ist, dass sich alle Neuronen nach gewisser Lernzeit im Zentrum der Oberfläche befinden. Bei Versuchen ohne zyklische Randbedingungen stellt sich dieses Verhalten sehr schnell (nach wenigen Lernzyklen) ein, und es können keinerlei Informationen mehr aus diesem „zentralen Neuronen-Cluster“ abgelesen werden.

Bei den vorherigen Versuchen mit dem Kohonen-Algorithmus wurde eine rechteckige SOM verwendet. Für die SOS ist es prinzipiell egal, welche Form die Oberfläche hat, da ja die Positionen der Neuronen frei wählbar sind. Eine ideale geschlossene Oberfläche wäre natürlich eine Kugeloberfläche [Zell97]. In diesem Versuch wurde allerdings die rechteckige Oberfläche gewählt und über die zyklischen Randbedingungen in sich geschlossen.
Bewegt man sich von einem Punkt auf der rechteckigen Oberfläche auf einen Rand zu und überquert diesen, so soll man am gegenüberliegenden Rand wieder auftauchen.

Das bedeutet wiederum, dass Punkte an gegenüberliegenden Rändern besonders benachbart sind. Die Ecken des Rechtecks finden sich somit sogar in einem Punkt wieder. Abbildung

3.45 veranschaulicht das Vorgehen für die Abszisse. Analoges gilt auch für die Ordinate. Berechnet wird die x-Distanz Δx_{jZ} vom Neuron c_j zum Erregungszentrum Z.

Die so geschlossene Oberfläche ist ein Torus. Neuronen können Anziehungskräfte auf Neuronen auf der anderen Seite des Rechtecks haben und sich über die Ränder des Rechtecks auf die gegenüberliegende Seite bewegen. Da die Neuronen nicht mehr an ein festes Neuronengitter, wie bei der Kohonenkarte gebunden sind, ist es kein Problem, Neuronen zu entfernen bzw. neu einzufügen. Ein neues Neuron braucht hierzu nur eine Startposition und einen Gewichtsvektor.

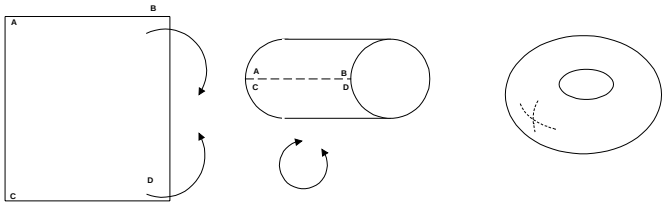

Abbildung 3.44: Schließen der offenen rechteckigen Kohonenkarte zu einem Torus.

Um die Dichteverteilung des Eingaberaumes besser zu modellieren, wurde der ursprüngliche SOS Algorithmus folgendermaßen modifiziert:
Pro Epoche wird das Neuron mit den meisten Treffern aus der Menge der Trainingsdaten ermittelt (mit einem Treffer ist hier gemeint, dass das Neuron einmal Siegerneuron war). Dieses Neuron ist ein guter Repräsentant für die meisten Trainingskurven.
Ebenso wird das Neuron ermittelt, das am wenigsten bzw. keinen Treffer erhält. Dieses Neuron ist keiner der Trainingskurven ähnlich und wird nun einfach in die Nähe des Siegerneurons gesetzt. Sein Gewichtsvektor wird mit dem Gewichtsvektor des Siegerneurons belegt.
Mit diesem Vorgehen wird erreicht, dass sich die Neuronendichte in der Nähe von Siegerneuronen (Erregungszentren) verstärkt. Neuronen, die außerhalb des Einflussbereichs solcher Zentren liegen, können somit auch erfasst werden.

Ein großer Vorteil des Verfahrens wird bei der Visualisierung offensichtlich. Während sich bei der Kohonenkarte hierbei durchaus Probleme ergeben können, wird hier einfach nur die Position der Neuronen dargestellt. Während des Trainings kann hierbei schon beobachtet werden, wie sich die Neuronen zu den Erregungszentren hinbewegen. Somit lassen sich die Cluster zusammengehöriger Trainingskurven besonders deutlich sichtbar machen.

Abbildung 3.45 zeigt die Ergebnisse von Trainingsläufen mit der SOS, wobei die Größe und Farbe der Neuronen anzeigt, wie viele Treffer das Neuron im letzten Durchlauf hatte. Für Anlage 1 wurden 100 Neuronen und als Eingabe die *RF_Match_Tuning_Position*-Signalverläufe verwendet. Die SOS Karte konvergiert wie erwartet in zwei Erregungszentren für die typischen Signalverläufe der Klassen von *„good“* bzw. *„bad“*. Für die *Emission_1*-Kurven von Anlage 2 bildet sich ein Hauptcluster heraus und zeigt damit, dass dieser Kurventyp auch am häufigsten in den Trainingsdaten vorhanden ist. Die restlichen Neuronen

haben sich nicht so deutlich wie beim vorherigen Beispiel gruppiert. Dennoch ist der Abstand dieser Neuronen untereinander geringer als der Abstand zum Hauptcluster.

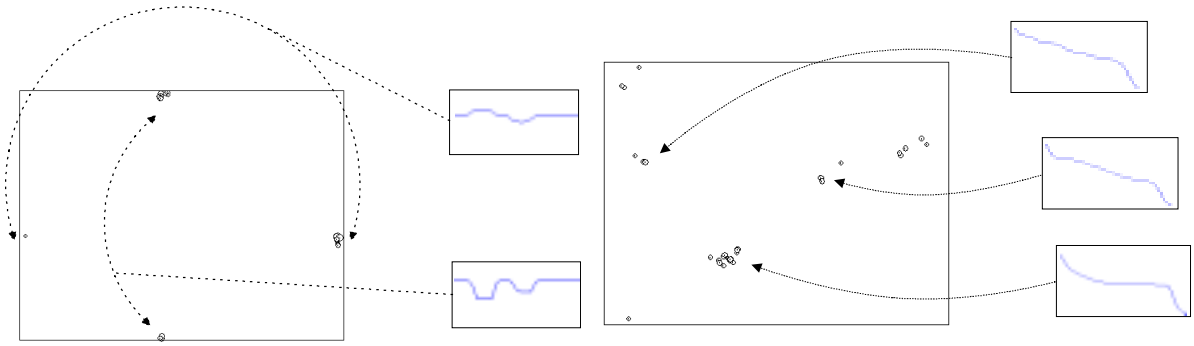

Abbildung 3.45: Links: 100 Neuronen auf zwei Cluster für zwei verschiedene Kurventypen der RF_Match_Tuning_Position. Die SOS Karte konvergiert wie erwartet in zwei Erregungszentren für die beiden trainierten Klassen. Rechts: 36 Neuronen mit einem Hauptcluster bei Emission_1-Parameterkurven.

3.2.6 Datenklassifizierung mit RBF-Netzen

In diesem Abschnitt werden RBF-Netze (Radial Basis Function) zur Extraktion klassifikationsrelevanter Kennzahlen zeitabhängiger Parameterkurven untersucht. In Kapitel 3.2.2 wurden hierzu Backpropagation-Netze untersucht. Es zeigte sich allerdings bei diesem Typ von neuronalen Netzwerken, dass es zu Fehlklassifikationen kommen kann. In Abbildung 3.49 ist die Trennung eines zweidimensionalen Eingaberaumes dargestellt. Die als Rauten dargestellte Muster entsprechen Testdaten, die zu den trainierten Daten stark unterschiedlich sind. Da sie aber dennoch in der Hälfte des Eingaberaumes mit dem Ausgabewert 0 liegen (Klasse *good*), werden sie vom Multilayer-Perzeptron als gute Daten interpretiert. Wünschenswert wäre an dieser Stelle natürlich eine Ausgabe des neuronalen Netzwerkes von 0,5, was einer dritten Klasse (unbekannt) entsprechen würde.

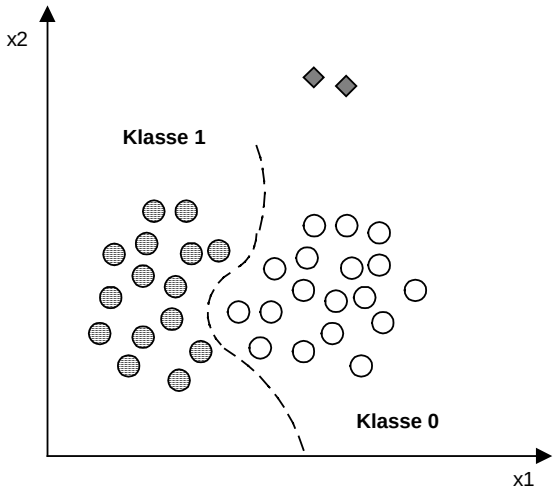

Abbildung 3.46: Trennung des Eingaberaumes in 2 Hälften durch ein Backpropagation-Netz mit einem Ausgabeneuron.

Da die Neuronen der verdeckten Schicht von RBF-Netzen anstelle einer sigmoiden Ausgabefunktion eine radiale Basisfunktion benützen, können sie dieses Problem der Backpropagation-Netzwerke umgehen. Dieses Problem könnte natürlich auch dadurch gelöst werden, indem einfach weitere Ausgabeneuronen beim Backpropagation-Algorithmus verwendet werden. Die Netzwerkkomplexität und damit auch der Trainingsaufwand nimmt dann allerdings erheblich zu.

RBF-Netze sind 3-schichtige, vorwärtsgerichtete, neuronale Netzwerke, die mit einem überwachten Lernalgorithmus trainiert werden. Die Neuronen in der verdeckten Schicht haben als Ausgabefunktion radialsymmetrische Basisfunktionen. Mathematisch gesehen werden diese Funktionen dazu benützt, die Übertragungsfunktion des Netzes zu approximieren und somit an die Trainingsdaten anzupassen. Für die Realisierung der RBF-Netze gibt es eine Vielzahl von Implementierungsvarianten (Wahl des Funktionstyps, Ermittlung der Funktionsstützstellen, Anpassung der Ausgabegewichte etc.). Im Folgenden werden die verwendeten Verfahren vorgestellt.

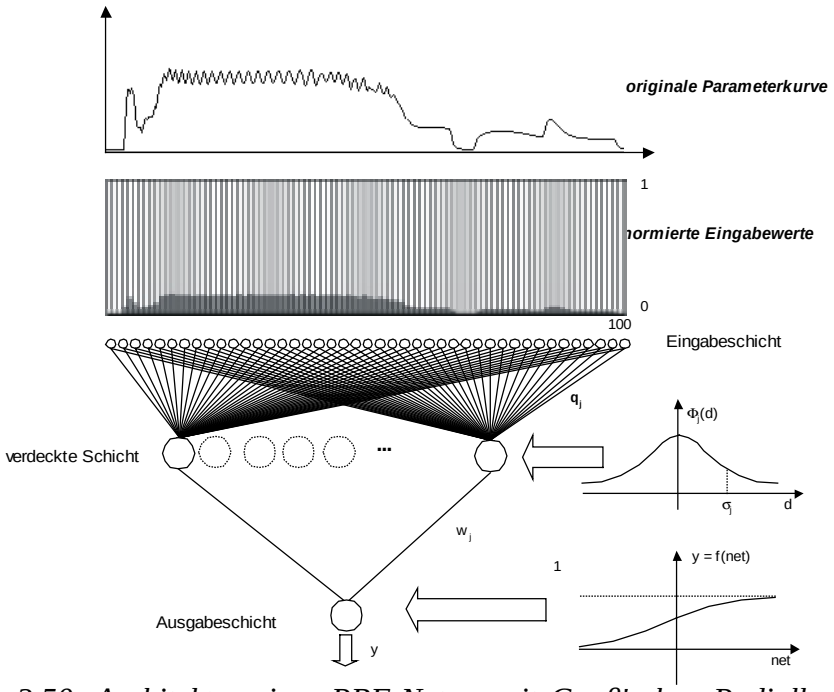

Abbildung 3.50: Architektur eines RBF-Netzes mit Gauß'schen Radialbasis-Funktionen und sigmoider Ausgabefunktion.

RBF-Netze bestehen aus 3 Schichten. An die Eingabeschicht werden die Trainings- bzw. Testvektoren angelegt. Ein Neuron c_j der verdeckten Schicht ermittelt über die euklidische Distanz d_j seines Referenzvektors $\boldsymbol{q}_j$ zu den Eingabedaten $\boldsymbol{x}$, und der Gaußfunktion als Aktivierungsfunktion, die Aktivität Φ_j. Die Breite der Gaußfunktion wird durch den Parameter σ_j angegeben und kann für jedes Neuron unterschiedlich sein.

$$d_j = \sqrt{\sum_i \left(x_i - q_{ij}\right)^2} \tag{3.38}$$

$$\Phi_j = e^{-\left(\frac{d_j}{\sigma_j}\right)^2} \tag{3.39}$$

Bei der verwendeten Netzarchitektur ist wie beim Backpropagation-Netzwerk aus Kapitel 3.2.3 nur ein Ausgabeneuron vorhanden. Jedes Neuron der verdeckten Schicht ist mit einem Ausgabegewicht w_j mit dem Ausgabeneuron verbunden.
Das Ausgabeneuron wird nun durch die Aktivität der Neuronen der verdeckten Schicht Φ_j und deren synaptischen Verbindungen w_j aktiviert. Da die Verwendung von sigmoiden Ausgabefunktionen die Klassifikationsleistung verbessert [Zell97], wird in diesem Kapitel die Darstellung der RBF-Netze (siehe 2.2.4) um die Verwendung einer sigmoiden Ausgabefunktion *f* erweitert.

Für die Aktivität *net* des Ausgabeneurons und die Netzwerkausgabe *y* ergibt sich:

$$net = \sum_j \Phi_j w_j \tag{3.40}$$

$$y = f(net) \tag{3.41}$$

In der verdeckten Schicht werden Neuronen, die dem Eingabemuster sehr ähnlich sind, besonders stark aktiviert. Neuronen, die eine größere euklidische Distanz zum Eingabemuster haben, werden nicht aktiviert. Die Verbindungsgewichte zum Ausgang können als Faktor für die Radialbasis-Funktionen gesehen werden. Ein negatives Gewicht entspricht somit einem „Tal“ (Ausgabe 0) in der Übertragungsfunktion, ein positives Gewicht erzeugt einen „Berg“ (Ausgabe 1). Wird kein Neuron der verdeckten Schicht aktiviert (weil das Eingabemuster weitab von den trainierten Mustern ist), so ist die Aktivierung des Ausgabeneurons 0 und somit sein Ausgabewert $y = 0{,}5$.

Im eindimensionalen Fall (1 Eingabeneuron) lässt sich der Zusammenhang zwischen den Radialbasis-Funktionen mit Stützstellen und deren Ausgabegewichten und der Übertragungsfunktion des Netzwerkes anschaulich zeigen. Dargestellt sind die Radialbasis-Funktionen von drei Neuronen im eindimensionalen Eingaberaum in Abhängigkeit vom Eingabewert *x*.

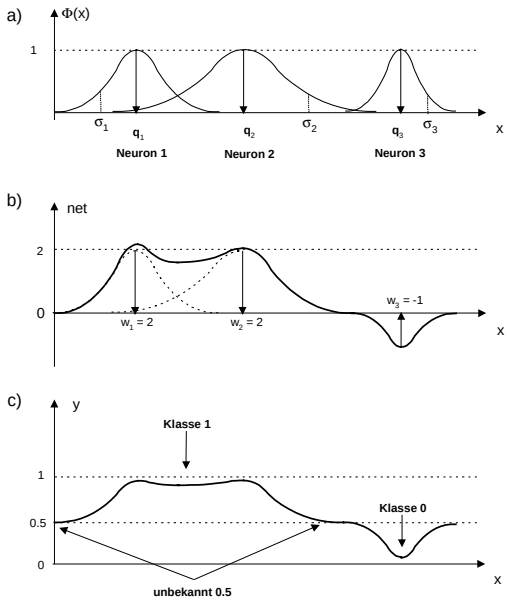

Abbildung 3.51: Übertragungsfunktion eines RBF-Netzes mit eindimensionaler Eingabe x. Bild a) zeigt die Ausgabewerte Φ_j der verdeckten Neuronen, in b) ist die Aktivierung net des Ausgabeneurons dargestellt und c) gibt schließlich den Ausgabewert y des Netzes an, wobei die Ausgabe auf Werte zwischen 0 und 1 begrenzt wird (sigmoide Funktion).

Entscheidend für den Erfolg des Verfahrens ist die Wahl der Referenzvektoren $\boldsymbol{q}_j$. Die im Kapitel 3.2.4 und 3.2.5 benutzten selbstorganisierenden Verfahren eignen sich gut, um Stützstellen zu ermitteln, die optimal im Eingaberaum verteilt sind. Als Referenzvektoren werden nur die Neuronen verwendet, die von den Testmustern aktiviert wurden. Die so ermittelten Neuronen werden direkt in die verdeckte Schicht des RBF-Netzes aufgenommen.

In diesem Kapitel steht Vorwissen über eine Klasseneinteilung der Parameterkurven zur Verfügung. Es kann damit ein Problem, das bei den selbstorganisierenden Verfahren besteht, umgangen werden. Werden dem Kohonen-Netz in einem Trainingslauf viele Muster der Klasse 0 präsentiert und nur sehr wenig Muster der Klasse 1, so kommt es dazu, dass im ungünstigsten Fall keines oder nur sehr wenige Neuronen sich auf die Muster der Klasse 1 einstellen. In der Realität ist dies allerdings die Regel, da einer großen Menge korrekter Prozessierungen nur eine kleine Anzahl Ausreißer gegenübersteht.
Die Lösung dieses Problems ist, die beiden Klassen getrennt mit einem Kohonen-Netz zu trainieren. Jeder Trainingslauf liefert dann optimale Stützstellen, sowohl für die Klasse „*good*“ als auch für die Klasse „*bad*“.

Analog zum Backpropagation-Algorithmus kann nun die Abweichung zwischen Zielwert *t* und Ausgabe *y* (Deltaregel) dazu benützt werden, die Radien σ_j der Radialbasis-Funktionen und die Ausgabegewichte w_j iterativ zu berechnen. Die Fehler der verdeckten Schicht werden äquivalent wie bei einem dreischichtigen Backpropagation-Netzwerk aus dem Fehler der Ausgabeschicht rückpropagiert.

Ziel des Trainings ist, alle freien Parameter des neuronalen Netzes (σ_j, w_j) so zu adaptieren, dass die Summe E der halben quadratischen Fehler E_p über alle Trainingspaare p möglichst minimal wird.

$$E = \sum_p E_p \tag{3.42}$$

$$E_p = \frac{1}{2}\sum_j \left(t_p - y_p\right)^2 \tag{3.43}$$

Die Fehlerfunktion kann man sich (im Fall von nur zwei Parametern) als Fläche vorstellen. Für eine bestimmte Wahl aller Parameter des neuronalen Netzes erhält man einen bestimmten Wert für E. Beim Start des Trainings befindet man sich also auf dieser Fehlerfläche mit Sicherheit auf einem „Berg“, da zu Beginn des Trainings der Fehler noch sehr groß ist.
Ist die Fehlerfläche stark zerklüftet, gibt es also viele lokale Minima, ist die Gefahr groß, dass das Netzwerk in einem dieser Minima hängen bleibt. Alle Parameter A werden so verändert, dass sie sich in Richtung des negativen Gradienten bewegen. η stellt dabei die Lernrate dar.

$$\Delta A = -\eta \nabla E(A) \tag{3.44}$$

Für einen bestimmten Parameter a_j gilt allgemein

$$\Delta a_j = -\eta \frac{\partial}{\partial a_j} E(A) = \sum_p -\eta \frac{\partial}{\partial a_j} E_p \tag{3.45}$$

Hierbei handelt es sich um das Batch-Trainingsverfahren, bei dem erst alle Muster p dem neuronalen Netz präsentiert werden, um anschließend alle freien Parameter in einem Schritt anzupassen.

In der Praxis wird hauptsächlich das *Online*-Trainingsverfahren eingesetzt, bei dem die Parameter nach jedem Trainingsmuster angepasst werden. Beim Backpropagation-Netzwerk aus Kapitel 2.3.3 sowie beim verwendeten RBF-Netz wurde das *Online*-Trainingsverfahren angewendet.

$$\Delta_p a = -\eta \frac{\partial}{\partial a_j} E_p \tag{3.46}$$

Abbildung 3.49 zeigt einen Schnitt durch eine denkbare Fehlerfläche entlang des Parameters a_1. Der Punkt gibt den Wert des Parameters a_1 und den Fehler, den das neuronalen Netz gemacht hat, zum Zeitpunkt t an. Zum Zeitpunkt $t+1$ soll sich der Parameter a_1 ein Bruchstück in Richtung des negativen Gradienten (Pfeil) bewegt und somit den Fehler E um ein Bruchstück minimiert haben.

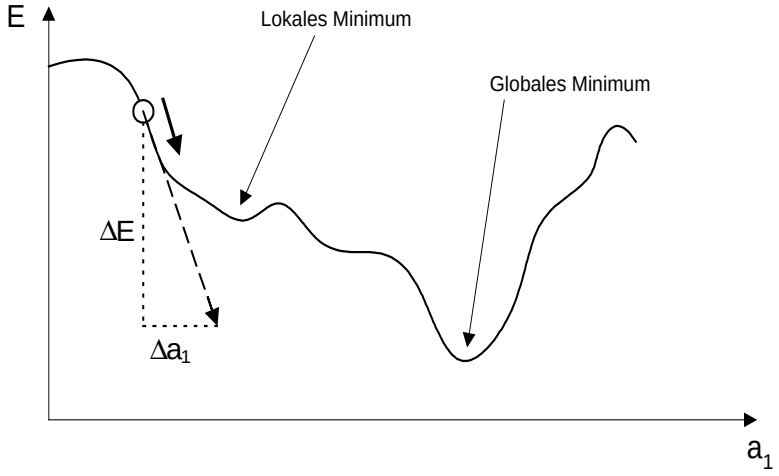

Abbildung 3.49: Fehlerfläche und Gradientenabstiegsverfahren.

Im Folgenden wird die Herleitung für die Formeln zur Berechnung der Gaußfunktionsradien σ_j und der Ausgabegewichte w_j speziell für die verwendete Architektur durchgeführt. Zudem wird die *Online*-Regel betrachtet und für Δ_p kurz Δ geschrieben.

Herleitung von $\Delta\sigma_j$ und Δw_j

Zuerst wird das Delta für die Änderung der Ausgabegewichte w_j bestimmt.

$$\Delta_p w_j = \Delta w_j = -\eta_1 \frac{\partial}{\partial w_j} E_p \tag{3.47}$$

Um die Fehlerfunktion abzuleiten, ist es nötig, mehrfach die Kettenregel anzuwenden. E_p ist bekannt und hängt pro Trainingsmuster p (*Online*-Lernregel) nur von der Ausgabe y des Netzwerkes ab.

$$\frac{\partial E_p}{\partial w_j} = \frac{\partial E_p}{\partial net} \cdot \frac{\partial net}{\partial w_j} \tag{3.48}$$

$$\frac{\partial E_p}{\partial net} = \frac{\partial E_p}{\partial y} \cdot \frac{\partial y}{\partial net} \tag{3.49}$$

Mit Formel (3.43) für E_p und der verwendeten Architektur (nur ein Ausgabeneuron) gilt somit:

$$\frac{\partial E_p}{\partial y_p} = -\left(t_p - y_p\right) \tag{3.50}$$

Die Aktivierung *net* des Ausgabeneurons ist die gewichtete Summe der Ausgabewerte Φ_j, der Neuronen c_j in der verdeckten Schicht:

$$net = \sum_j w_j \Phi_j \tag{3.51}$$

Für die Ableitung von *net* gilt damit:

$$\frac{\partial net}{\partial w_j} = \Phi_j \tag{3.52}$$

Die Ausgabe *y* wird mit der sigmoiden Funktion ermittelt:

$$f(net) = y = \frac{1}{1+e^{-net}} \tag{3.53}$$

Die Ableitung von *f(net)* nach *net* ist:

$$\frac{\partial y}{\partial net} = f'(net) = y(1-y) \tag{3.54}$$

Nun sind alle Ableitungen berechnet und die Formel für Δw_j kann angegeben werden:

$$\Delta w_j = \eta_1 (t_p - y) y (1-y) \Phi_j \tag{3.55}$$

Hierbei soll der Fehler am Ausgabeneuron des Netzwerkes (analog wie bei Backpropagation in Kapitel 3) als δ_{out} bezeichnet werden:

$$\delta_{out} = (t_p - y) y (1-y) \tag{3.56}$$

Somit kann Δw_j kürzer angegeben werden als:

$$\Delta w_j = \eta_1 \delta_{out} \Phi_j \tag{3.57}$$

Analog lässt sich auch die Formel für σ_j berechnen.
Die Aktivität Φ_j der Neuronen der verdeckten Schicht ist eine Gaußfunktion in Abhängigkeit von σ_j:

$$\Phi_j = e^{-\left(\frac{d}{\sigma_j}\right)^2} \tag{3.58}$$

Folgender Ansatz ist nun zu machen:

$$\frac{\partial E_p}{\partial \sigma_j} = \frac{\partial E_p}{\partial \Phi_j} \cdot \frac{\partial \Phi_j}{\partial \sigma_j} \tag{3.59}$$

Die mehrfache Anwendung der Kettenregel führt dann zu folgenden Ableitungen:

$$\frac{\partial E_p}{\partial \Phi_j} = \frac{\partial E_p}{\partial net} \cdot \frac{\partial net}{\partial \Phi_j} \tag{3.60}$$

$$\frac{\partial E_p}{\partial net} = \frac{\partial E_p}{\partial y} \cdot \frac{\partial y}{\partial net} \tag{3.61}$$

Hierbei sind nur noch folgende Ableitungen zu berechnen, die restlichen sind bereits von der Herleitung für Δw_j bekannt.

Mit Gleichung (3.51) folgt für die Ableitung von *net:*

$$\frac{\partial net}{\partial \Phi_j} = w_j \tag{3.62}$$

Mit Gleichung (3.58) folgt für die Ableitung von Φ_j*:*

$$\frac{\partial \Phi_j}{\partial \sigma_j} = e^{-\left(\frac{d}{\sigma_j}\right)^2} 2\frac{d^2}{\sigma_j^3} = 2 \cdot \Phi_j \frac{d^2}{\sigma_j^3} \tag{3.63}$$

Der Faktor 2 wird zur Lernrate η hinzugerechnet, somit kann die Formel für die Berechnung von $\Delta\sigma_j$ angegeben werden als

$$\Delta\sigma_j = \eta_2 \Phi_j \frac{d^2}{\sigma_j^3} w_j (t_p - y) y(1-y) \tag{3.64}$$

Die Formel zeigt, dass analog wie bei Backpropagation der Fehler der Ausgabeschicht (des Ausgabeneurons) über die Gewichte w_j zu den Neuronen der verdeckten Schicht rückpropagiert werden kann.
Der Fehler δ_j eines Neurons c_j der verdeckten Schicht kann somit definiert werden als

$$\delta_j = w_j (t_p - y) y(1-y) = w_j \delta_{out} \tag{3.65}$$

und die Formel zur Berechnung von $\Delta\sigma_j$ kürzer angegeben werden als

$$\Delta\sigma_j = \eta_2 \frac{d^2}{\sigma_j^3} \Phi_j \delta_j \tag{3.66}$$

Diese Formeln können nun im *Feedbackward*-Schritt des Lernalgorithmus angewendet werden, um die Gaußradien und die Ausgabegewichte anzupassen. Der Fehler des Ausgabeneurons kann wie beim Backpropagation-Netzwerk von der Ausgabeschicht zu der verdeckten Schicht zurückgerechnet werden.

Die Neuronen im RBF-Netz ermitteln die Ausgabewerte mit den zugehörigen Radialbasis-Funktionen. Diese können auch als lokale rezeptive Felder pro Neuron angesehen werden. Im Gegensatz zum Backpropagation-Netzwerk nimmt also ein Neuron im RBF-Netz keine Trennung des Eingaberaumes in zwei Hälften vor. Vielmehr bildet es (bei Verwendung der Gaußfunktion) eine Sphäre um das Zentrum der Radialbasis-Funktion

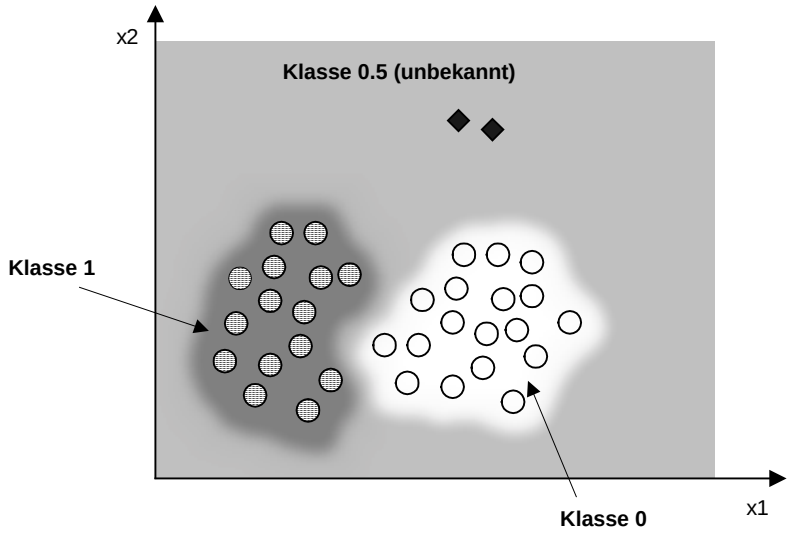

Abbildung 3.53: Aufteilung des Eingaberaumes in zwei Klassen {0, 1}. Für umliegende Bereiche gibt das RBF-Netz den Ausgabewert 0,5 aus.

Die Aufsummierung dieser Gauß'schen Sphären am Ausgabeneuron führt zur Entstehung von Gebieten für die Klassen „*good*“ oder „*bad*“ mit den Ausgabewerten nahe bei 0 oder 1. Zwischen diesen Gebieten, bzw. weit entfernt davon, gibt das neuronale Netz Werte um 0,5 aus, was einer dritten Klasse „unbekannt“ entspricht. Der Ausgabewert gibt zudem die Zugehörigkeitswahrscheinlichkeit zu einer der drei Klassen an, es wird also keine „scharfe“ Zuordnung zu einer Klasse (wie bspw. beim RCE-Netz) gemacht. Das Problem der Fehlklassifikation beim Backpropagation-Netzwerk (Kap 3.2.3) tritt beim RBF-Netz nicht auf. Liegen Testmuster (als Rauten dargestellt) außerhalb der trainierten Muster, so gibt das RBF-Netz den Wert 0,5 (unbekannt) aus.

Beim Training zeigte sich, dass, im Vergleich zum Backpropagation-Netzwerk, der Fehlerverlauf stark schwankt. Die Ursache liegt vermutlich darin begründet, dass Gaußradien und Ausgabegewichte gemeinsam einen Weg finden müssen, um den Fehler zu minimieren. So kommt es vor, dass eine Änderung für den einen Parameter eine momentane Verschlechterung für einen anderen Parameter bewirkt.

Um die Fehlerfunktion zu glätten, wurde deshalb zunächst für die Änderung der Parameter σ_j und w_j ein Momentum-Term eingeführt (siehe Kapitel 3.2.3). Eine zeitlich veränderliche Lernrate $\eta(t)$ konnte zusätzlich noch zur Glättung der Fehlerfläche beitragen. Mit der Lernrate η und dem Momentum-Faktor α wurden schließlich folgende Momentum-Terme implementiert

$$\Delta w_j(t+1) = \eta_1\left((1-\alpha_1)\cdot\delta_{out}\Phi_j + \alpha_1\Delta w_j(t)\right) \tag{3.67}$$

$$\Delta\sigma_j(t+1) = \eta_2\left((1-\alpha_2)\cdot\frac{d^2}{\sigma_j^3}\Phi_j\delta_j + \alpha_2\Delta\sigma_j(t)\right) \tag{3.68}$$

Diese Formeln berücksichtigen auch, dass bei kleiner werdenden Lernraten auch das Momentum sinkt. Bei konstanten α und η können diese Formeln leicht in die bereits bekannte Momentumsformel (Kapitel 3.2.2, Gleichung (3.16)) umgerechnet werden.

Da die Klasseneinteilung der *Emission_1*-Parameterkurve hauptsächlich auf die Oszillationen am *Overetch* zurückzuführen sind, wurde die Klassifikationsleistung von Backpropagation- und RBF-Netzen für Sinuskurven untersucht. Als „*good*“ wurde eine Gerade $y=1$, als „*bad*“ mit weißem Rauschen überlagerte Sinuskurven $y=sin(x)+\gamma$, und als Testdaten phasenverschobene Sinus-Kurven $y=sin(x+\varphi)$ verwendet (siehe Tabelle 3.13). Ab einer Phasenverschiebung von $\pi/3$ kann das RBF-Netzwerk die Oszillation nicht mehr als solche klassifizieren.
Für den Einsatz dieses neuronalen Netzwerktypus heißt das, dass zum Training alle als schlecht aufgezeichneten Kurvenverläufe mit Rauschen bzw. Phasenverschiebungen versehen und dem Netz trainiert werden müssen. Zumindest gibt das RBF-Netz keine falsche Klassifizierung, sondern klassifiziert die phasenverschobenen Kurven mit einem Wert nahe 0,5, was der Klasse „unbekannt“ entspricht.

Tabelle 3.13: Klassifikationsleistung bei phasenverschobenen Sinuskurven

	$\varphi=0$	$\varphi=\pi/6$	$\varphi=\pi/3$	$\varphi=\pi/2$
Output$_{Backprob}$	0.93	0.9	0.63	0.1
Output$_{RBF}$	0.97	0.89	0.66	0.5
Solloutput	1	1	1	1

3.2.7 Diskussion

Ziel der Simulationen und Experimente in diesem Kapitel war, neuronale Netze zur Kennzahlenextraktion, Modellierung und Analyse nichtlinearer multisensorischer Zeitreihen, wie sie während der Prozessierung von Si-Wafer aufgezeichnet werden, zu untersuchen.
Zum verwendeten Datenmaterial war, im Gegensatz zu den Zeitreihen aus Kapitel 3.1, die Information vorhanden, dass gewisse Merkmale auf den Zeitreihen eine schlechte Prozessierung indizieren. Konkret indizieren z.B. Oszillationen auf dem *Overetch*-Kurvenverlauf in der *Emission_1*-Parameterkurve, dass die Ätzung nicht optimal verlaufen ist und Aluminiumreste auf den Scheiben zurückbleiben. Diese Reste beeinflussen die nachgeschalteten Prozessierungen und sind die Ursache für eine geringe Ausbeute der einzelnen Si-Wafer.
Nach der Vorverarbeitung und Analyse mit Verfahren der nichtlinearen Zeitreihenanalyse wurden Backpropagation-Netze zur Kennzahlenextraktion und Parameterselektion, Kohonen Self Organizing Feature Maps (SOM) und Self Organizing Surfaces (SOS) zur automatischen Klasseneinteilung und RBF-Netze (Radial Basis Function) zur Datenklassifikation verwendet.

Die Auswertungen mit Methoden der nichtlinearen Zeitreihenanalyse zeigen, dass Autokorrelationen in den Datenreihen enthalten sind und eine ausreichend hohe Samplingrate gewählt wurde. Kreuzkorrelationen zwischen den einzelnen Signalverläufen sind ebenfalls vorhanden und können teilweise zur Reduktion des Datenraumes verwendet werden. Aufgrund der Begrenzung der Schnittstellenkapazität sollten möglichst unabhängige Parameter erfasst werden, um so die zugrundeliegende Dynamik optimal modellieren zu können.
Zur Abschätzung der Komplexität bzw. der Dimensionalität für die Modellierung wurde die Einbettungsverzögerung τ_E unter Verwendung der Methode der *Average Mutual Information* und Dimensionalität des rekonstruierten Attraktors mit der *false-nearest-neighbor*-Methode bestimmt. Diese Abschätzungen können als Grundlage zum Netzwerkdesign verwendet werden [Kulkarni97]. In den untersuchten Zeitreihen liegt die Einbettungsverzögerung bei 13 Sekunden und die Einbettungsdimension im Bereich zwischen 2 und 6 Dimensionen.

Die anschließenden Auswertungen mit Backpropagation-Netzwerken mit einer verdeckten Schicht und einem Ausgabeneuron zeigen, dass diese Netze gut geeignet sind, die verschiedenen Zeitreihenverläufe aufgrund der *a-priori*-Klasseneinteilung zu separieren. Ein großer Vorteil dabei ist, dass zwar mit einer digitalen Klasseneinteilung trainiert wird, aber aufgrund der unterschiedlichen Ausprägung des Problems das Netzwerk in der Lage ist, die Klassenteilung in einen kontinuierlichen Bereich zu projizieren. Die gut/schlecht-Aussage relativiert sich damit zu einer Art Fuzzy-Logik für die Wahrscheinlichkeit der Klassenzugehörigkeit und bieten dem Anwender eine gute Entscheidungshilfe. Problematisch ist allerdings, dass aufgrund der Separation des Phasenraumes durch Hyperebenen es unter gewissen Umständen zu Fehlklassifikationen kommen kann, wie im Beispiel mit den Sinuskurven im Abschnitt 3.2.6 gezeigt wird.

Zusätzlich zur ursprünglichen Aufgabenstellung eröffnete sich ein Verfahren zur Analyse der Trainingsdaten zur Untersuchung auf deren Relevanz für das zu untersuchende Problem. Diese Option ist insofern interessant, da die Prozessanlagen mehrere hundert Parameter potentiell erfassen könnten, aufgrund der Einschränkungen in der Datenübertragung allerdings eine Selektion vorgenommen werden muss. Darüber hinaus sind viele Prozessvariablen noch nicht richtig verstanden. Die Information über die Relevanz bestimmter Parameter zur Fehlerdetektion ist eine Voraussetzung für den Erfolg nachgeschalteter Fehlererkennungs- bzw. Klassifikationsalgorithmen. Durch die Analyse der Synapsen des neuronalen Netzes können sowohl die Trainingsdaten als auch bestimmte Kurvenabschnitte beurteilt werden. Dies kann dazu eingesetzt werden, um ausgehend von der Information, ob eine Prozessierung normal oder abnormal ist, die verantwortlichen Parameter und *Recipe-Steps* aus den Prozessierungsdaten zu ermitteln.
Nachteil beider Verfahren ist, dass die Ergebnisse des Netzwerkes (somit auch der Parameter Analyse) stark von der Auswahl der Trainingsdaten abhängig sind. Unterschiedliche Trainingsdaten können zu unterschiedlichen Ergebnissen führen. Fehlerhafte Daten (schlecht aufgezeichnete Daten) können die Ergebnisse verfälschen und müssen vom Training ausgeschlossen werden.

Anschließend wurden selbstorganisierende Verfahren wie SOM und SOS zur Analyse der Daten verwendet. Anders als in Kapitel 3.1.3 wurden nicht durch PCA vorkomprimierte Daten, sondern die Originalkurven dem Netzwerk trainiert. Zusätzlich stand die Information über die Signifikanz bestimmter Kurvenabschnitte und Verläufe zur Verfügung.
Der klassische Kohonen-Algorithmus funktioniert sehr stabil. Die Parameter sollten lediglich so eingestellt sein, dass sich die Kohonenkarte vollständig ausprägen kann. Selbst wenn dies nicht der Fall ist, gibt es immer noch Bereiche der SOM, die sich korrekt adaptiert haben.
Durch die Abwandlung des Kohonen-Algorithmus ist eine selbstorganisierende Oberfläche entstanden. Die Neuronen sind hierbei an kein festes Gitter mehr gebunden und können sich frei bewegen. Neuronen, deren Gewichte ähnliche Kurven repräsentieren, ziehen sich an und bilden Gruppen auf der SOS. Ein häufiges Problem bei SOM, das der Visualisierung von Clustergrenzen, ist somit hinfällig, da diese Gruppen bereits die gesuchten Cluster darstellen. Das Verfahren ist aber nicht so stabil wie der ursprüngliche Algorithmus von Kohonen. Sind die Parameter des Netzwerkes nicht richtig gewählt (Beweglichkeit der Neuronen zu hoch, Lernrate zu hoch, Startradius der Nachbarschaftsfunktion zu hoch), so können sich alle Neuronen zu einer Gruppe vereinen. Sind diese Parameter zu niedrig gewählt, so ergeben sich keine deutlichen Gruppierungen der Neuronen.

RBF-Netze reagieren dank ihres lokalen rezeptiven Feldes nur auf Testmuster, die ähnlich zu den Referenzvektoren, und damit zu den Trainingsmustern, sind. Die Wahrscheinlichkeit einer Fehlklassifikation ist bei RBF-Netzen geringer, da diese die präsentierten Testmuster vorsichtiger bewerten als z.B. Backpropagation-Netze. Für ein den Trainingsdaten sehr unähnliches Muster kann es bei Backpropagation vorkommen, dass dieses Muster eindeutig einer Klasse zugeordnet wird (weil es in einer Hälfte des getrennten Musterraums liegt), wogegen ein RBF-Netz an dieser Stelle das Muster als unbekannt einstufen würde.

Dank seines einfachen Aufbaus kann mit dem RBF-Netz sehr flexibel gearbeitet werden. So ist es kein Problem, Neuronen aus der verdeckten Schicht, je nach Bedarf, zu entfernen oder hinzuzufügen.
Für die Wahl der Stützstellen können verschiedenste Verfahren eingesetzt werden. Das hier verwendete SOM-Verfahren ist bereits ziemlich aufwendig und könnte im Hinblick auf kürzere Trainingszeiten durch einfachere Methoden ersetzt werden. Die mächtigste Verbesserung wäre der Schritt vom RBF-Netz (Radial Basis Function) zum HBF-Netz (Hyper Basis Function). Bei dieser Erweiterung werden dann keine radialsymmetrischen Funktionen mehr benützt, sondern in jeder Dimension (ideal anzupassende) unterschiedlich breite Funktionen. Aus den Sphären des RBF werden z. B. Ellipsoide. Als mächtigste Variante der HBF-Netze könnten schließlich noch ideal gedrehte Ellipsoide verwendet werden. Für die gezeigten Zeitreihen erreichten die untersuchten Methoden eine ausreichende Sensibiliät und erbrachten eine ausreichend hohe Separations- und Klassifikationsleistung.

3.3 Mustererkennung und Klassifizierung hochdimensionaler Testdatenfelder

In den Kapiteln 3.1 und 3.2 wurden neuronale Netzwerkmodelle zur Kennzahlenextraktion und Dimensionsreduzierung multisensorischer Zeitreihen für die anschließende Fehlererkennung und Diagnose untersucht. Es stellte sich dabei heraus, dass bereits eindimensionale Fehlererkennungsalgorithmen viele Abnormitäten detektieren, allerdings auch multivariate Effekte in den Datenräumen auftreten.
Ziel dieses Kapitels ist, verschiedene Methoden zur Mustererkennung und Klassifizierung hochdimensionaler Datenräume zu untersuchen. Es werden hier, im Gegensatz zu den vorherigen Kapiteln, Testdatensätze prozessierter Si-Wafer verwendet, die nicht während der Prozessierung, sondern im Anschluss, nach Fertigstellung im *Frontend*, gemessen werden. Die Daten sind elektrische Daten (PCM-Messungen), die vor der eigentlichen Funktionsmessung und Ausbeutebestimmung zur Vorselektion aufgenommen werden, sowie die für die Klasseneinteilung verwendete Waferausbeute (siehe Kapitel 1). Die Ergebnisse gelten generell, können also auch für die Klassifizierung der multisensorischen Daten verwendet werden, falls Vorwissen über eine Klasseneinteilung vorhanden ist.
Nach der Datenvorverarbeitung, werden wachsende und überwacht wachsende neuronale Gase zur Bestimmung von Referenzvektoren für z.B. RBF-Netze untersucht. Als Referenz für die Klassifikationsalgorithmen wird der *k-next-neighbor*-Algorithmus herangezogen, der im anschließenden Kapitel benützt wird. Zur Datenklassifikation werden die lokal konstanten Abbildungen, die wachsenden lokal konstanten Abbildungen und RBF-Netze verwendet. Im abschließenden Kapitel werden die Ergebnisse diskutiert.

3.3.1 Datenvorverarbeitung

Das wichtigste Ziel einer Vorverarbeitung ist oft die Reduktion der Dimension der Rohdaten. Begründet ist dieses Ziel durch den „curse of dimensionality“. In den meisten Fällen steigt die Rechenzeit für die Simulation eines Problems insbesondere bei Verwendung neuronaler Netze nicht linear, sondern eher exponentiell mit der Dimension. So kann eine Dimensionsreduzierung trotz des unvermeidbaren Informationsverlustes die Effizienz eines Systems steigern. Betrachtet man höherdimensionale Datensätze, so ist es außerdem wahrscheinlich, dass die inhärente Dimensionalität des Problems geringer ist als die reine Anzahl an Parametern. In einem physikalisch relevanten Fall, wie den in diesem Kapitel untersuchten Waferdaten, wäre das beispielsweise gegeben, wenn an einem ohmschen Widerstand sowohl Spannung als auch Stromstärke gemessen wird. Die inhärente Dimensionalität dieses zweidimensionalen Datensatzes wäre dann gleich Eins.

In den vorliegenden Daten wurden besonders kritische Parameter selektiert. Nach der Erfahrung des zuständigen Produktingenieurs reagiert die Ausbeute des zugehörigen Wafers besonders sensibel auf 22 von insgesamt 65 Messwerten. In den folgenden Untersuchungen wurden daher nur diese 22 verwendet.
Diese Rohdaten repräsentieren meist physikalische Größen, sind mit einer Einheit behaftet und werden in verschiedenen Skalen, z.B. logarithmisch, gemessen. Sämtliche in dieser Arbeit behandelten Verfahren basieren, wie auch klassische statistische Maßnahmen, auf einem mathematisch einfachen Ähnlichkeitsmaß, normalerweise der euklidischen Distanz. Daher sind unterschiedliche Skalierungen und Einheiten der einzelnen Messwerte insbesondere aber auch verschiedene Größenordnungen durchaus kritisch.

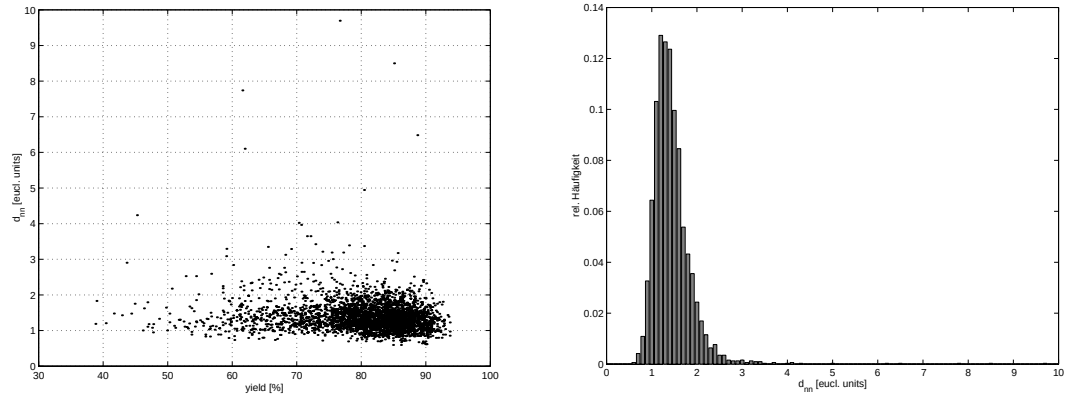

Abbildung 3.54: Verteilung der Abstände der Waferdaten zum jeweils euklidisch nächsten Muster im Eingaberaum. Die Datenpunkte haben einen minimalen Abstand voneinander und es sind einige Ausreißer zu erkennen.

Bei einem MOSFET z.B., der einen Verstärkungsfaktor von 100 und eine Einsatzspannung von 50 mV hat, kann je nach Spezifikation die Schaltung völlig kritisch, oder aber unempfindlich gegenüber Schwankungen in diesen Werten sein. So sind die Anforderungen an einen FET höher, wenn dieser als Verstärker eingesetzt wird, als als Teil einer dynamischen Speicherzelle. Im Extremfall sind sich also zwei Transistoren „ähnlicher“ (auf die Wafer bezogen wäre dieses Ähnlichkeitsmaß die Ausbeute), wenn ihre Verstärkungsfaktoren um 50 differieren, als wenn sich ihre Einsatzspannungen nur um 1 mV unterscheiden. Diese Information führt in der Waferproduktion derzeit zu einer univariaten Grenzwertprüfung der PCM-Parameter. Liegen zu viele PCM-Werte eines Wafers außerhalb der Spezifikationsgrenzen, wird er nicht zur Funktionsmessung zugelassen und aussortiert.

Da man *a-priori* nicht feststellen kann welche Größen und welche Schwankungen in den einzelnen Werten signifikant sind, ist es sinnvoll, die einzelnen Messgrößen zunächst gleich zu bewerten. Als erster Schritt werden alle Parameter, die in logarithmischen Skalen gemessen sind, durch Exponenzieren auf lineare Skalen projiziert. Jeder Wafer stellt mit seinen 22 PCM-Parametern einen Vektor im $\mathbb{R}^{22}$ dar. Dieser Satz an Messwerten wird für jede Dimension auf Mittelwert Null und Standardabweichung gleich Eins normiert.
Prinzipiell erwartet man einen Zusammenhang zwischen den PCM-Parametern und den zugehörigen Ausbeuten, d.h. Wafer mit ähnlichen Messwerten sollten auch ähnliche Ausbeuten aufweisen. Ohne diesen funktionellen Zusammenhang wird kein generalisierungs-

fähiger Algorithmus in der Lage sein, neue Muster korrekt zu klassifizieren. Schließlich beruhen alle klassischen Methoden wie auch neuronale Netze auf Ähnlichkeitsmaßen, meist dem euklidischen Abstand.

Für das vorliegende Datenmaterial wird deshalb zuerst die Voraussetzung für eine erfolgversprechende Klassifikation untersucht. Für jedes der 3122 Muster wird dasjenige Muster gesucht, welches den geringsten euklidischen Abstand hat, d.h. welches ihm am ähnlichsten ist. Dieser nächste Nachbar nn_i ist bestimmt durch:

$$nn_i = \min_{\mathbf{x}_j} \left\| \mathbf{x}_j - \mathbf{x}_i \right\| \tag{3.69}$$

Es zeigt sich zunächst, dass alle Eingabemuster einen minimalen Abstand voneinander haben (siehe Abbildung 3.54). Weiterhin sind einige Ausreißer zu erkennen, die weit von allen anderen Mustern entfernt sind.

Im nächsten Schritt wird die mittlere Distanz der Wafer bezogen auf den nächsten Nachbarn in der Ausbeute bestimmt. Die Waferausbeute wird normalerweise in Prozent angegeben. Um keine Verwirrung zwischen relativen Angaben und absoluten Waferausbeuten zu erzeugen, wird in diesem Kapitel auf die Angabe von Prozent bei der Waferausbeute verzichtet.

$$\Delta\bar{d} = E(\Delta d) = \frac{1}{N}\sum_{i=1}^{N} \Delta d_i \tag{3.70}$$

mit $\Delta d_i = d_i - d_{nn_i}$

Dieser mittlere Abstand beträgt $\Delta\bar{d} = 3{,}82$. Teilweise sind die Ausbeuten zu ähnlichen Mustern gehöriger Wafer bis zu 40 Prozent verschieden (siehe Abbildung 3.55)

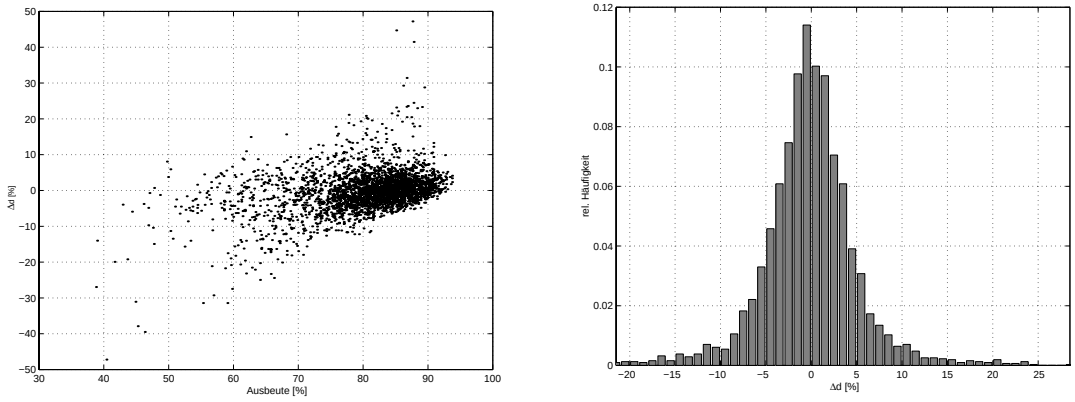

Abbildung 3.55: Verteilung der Differenzen in der Ausbeute aller Muster zu ihrem jeweiligen nächsten Nachbarn. Im Mittel weichen die Ausbeuten benachbarter Wafer um 3,82 ab. Für etwa 5% der Wafer beträgt diese Abweichung 10 oder mehr.

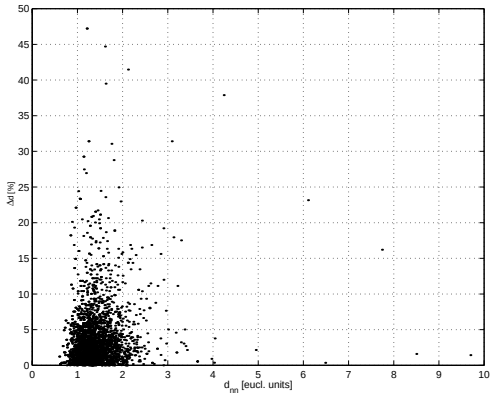

Abbildung 3.56: Korrelationsplot der beiden oberen Abbildungen 3.54 und 3.55. Dargestellt ist der Abstand zum nächsten Nachbarn gegen den Betrag der Differenz in der Ausbeute. Offensichtlich stammen große Abstände in der Ausbeute nicht nur von Ausreißern. Es ist keine Korrelation zu erkennen.

Abbildung 3.56 zeigt die Korrelation von euklidischem Abstand und Ausbeute-Differenz nächster Nachbarn. Ein Zusammenhang zwischen Abstand nächster Nachbarn und der Differenz in deren Ausbeute ist nicht zu erkennen. Im Gegenteil, viele Muster, die sich euklidisch nahe sind, sind in ihrer Ausbeute bis zu 50 Prozent verschieden. Ein gewisser Anteil, der derzeit allerdings nicht bekannt ist, liegt sicherlich in systematischen Messfehlern begründet. Es ist ebenso denkbar, dass die gesuchte Zuordnung von PCM-Parametern zur Ausbeute, aufgrund von lokalen Effekten auf dem Wafer, nur beschränkt gelten. Zur Bewertung der Klassifikationsleistung kann dieser mittlere Abstand von 3.8 herangezogen werden.

3.3.2 Wachsendes neuronales Gas (GNG)

Das wachsende neuronale Gas wird in erster Linie verwendet, um Stützstellen für spätere Klassifizierungsmaßnahmen zu erhalten d.h. insbesondere die Bestimmung von Zentren für lokal konstante bzw. lineare Abbildungen wie auch für RBF-Netze.

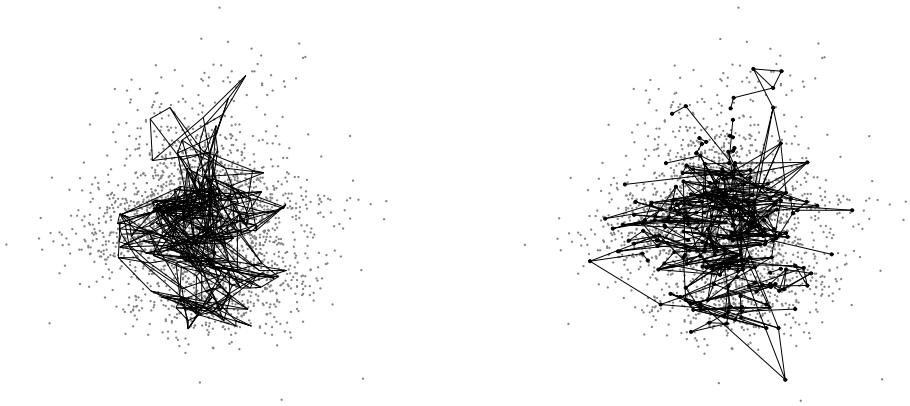

Abbildung 3.57: Vergleich einer quadratischen 17x17 SOM nach 20000 Trainingszyklen mit einem GNG, das auf die gleiche Größe von 289 Neuronen gewachsen ist, in der gleichen Projektion. Man sieht bereits hier wie das GNG sozusagen mehr Platz in Anspruch nimmt und sich der Topologie des Eingaberaumes besser anpasst.

Allerdings eignet sich dieser Algorithmus auch hervorragend zur Visualisierung hochdimensionaler Datenstrukturen. Während des Trainingsprozesses erkennt das GNG selbständig Cluster im hochdimensionalen Raum und erlernt die Topologie der Datenverteilung. Das GNG zeigt schon in einer zweidimensionalen Projektion eine deutlich bessere Abbildung des Eingaberaumes (siehe Abbildung 3.57) als eine vergleichbare SOM.

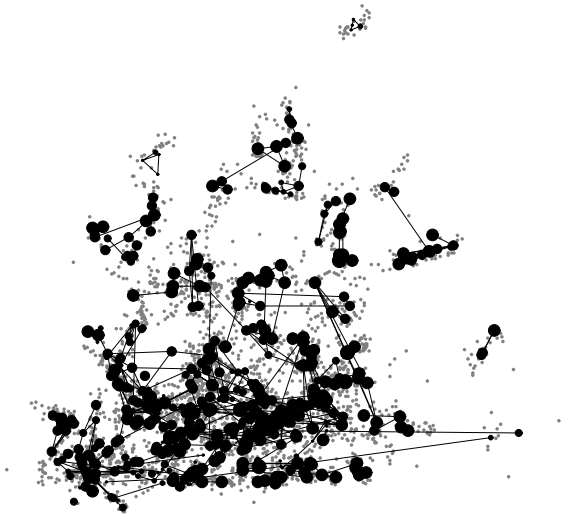

Abbildung 3.58: Zweidimensionale Projektion der Daten und der Neuronen eines wachsenden neuronalen Gases nach dem Einfügen von 400 Neuronen. Die Größe eines Neurons ist gegeben durch die mittlere Ausbeute der Wafer in seinem Voronoigebiet. Hier ist gut zu sehen, wie die Ausbeute mit der Position im Eingaberaum korreliert. Betrachtet man die einzelnen Cluster so ist deutlich zu sehen, dass benachbarte Neuronen ähnlich groß sind, d.h. ähnlichen mittleren Ausbeuten entsprechen.

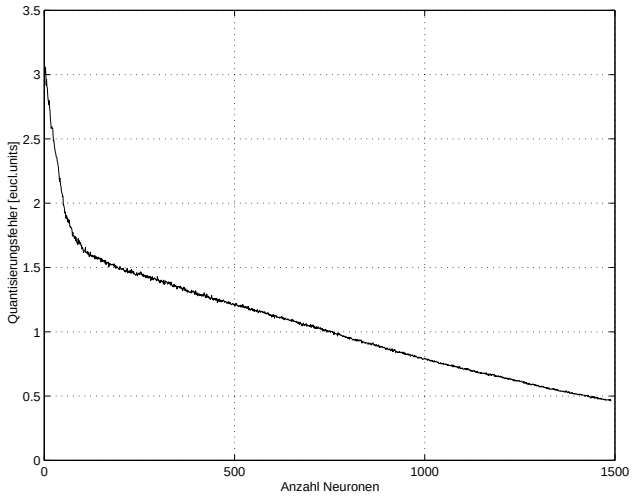

Abbildung 3.59: Verlauf des gleitenden Durchschnitts des Quantisierungsfehlers während des Trainings eines wachsenden neuronalen Gases. Dieser ist gegeben durch den mittleren Abstand der präsentierten Muster zum jeweiligen Gewinner. Die Form dieser Kurve weist auf zwei unterschiedliche Phasen während des Trainings hin. Sie ist typisch und unabhängig vom untersuchten Datensatz. Während der ersten Phase erfasst das GNG die globale Struktur des Datenraums während in der zweiten Phase der Raum feiner parzelliert wird.

Mathematisch äußert sich dies im mittleren Quantisierungsfehler, dem mittleren Abstand der Muster zu ihrem jeweiligen Gewinner. Er beträgt 1,64 für die SOFM und 1,4 für das GNG. Selbst bei einer Größe von 227 Einheiten (das entspricht der Anzahl an Neuronen in der SOFM, die nicht tot sind) beträgt der Fehler für das GNG nur 1,49.
Abbildung 3.59 zeigt den gleitenden Durchschnitt des momentanen lokalen Quantisierungsfehlers während des Trainings eines wachsenden neuronalen Gases. Dieser Quantisierungsfehler ist der Abstand des momentan präsentierten Musters vom Gewinnerneuron. Da während des Wachstumsprozesses das GNG den Raum immer genauer erschließt, fällt dieser Fehler.

Ein wachsendes neuronales Gas bildet eine Topologie-erhaltende Karte des Eingaberaumes in beliebiger Genauigkeit [Bruske95]. Kanten werden zwischen Neuronen nur in solchen Regionen gebildet, wo die Wahrscheinlichkeitsdichte ungleich Null ist. Dieser Algorithmus kann damit auch zum Auffinden von Clustern in hochdimensionalen Datenräumen verwendet werden. Für ein Neuron c_i werden alle Nachbarn n_i, also Neuronen die sich von diesem aus über beliebig viele Kanten erreichen lassen, betrachtet. Dann wird als Cluster $C_i = \bigcup V_{ni}$ die Vereinigung aller Voronoimengen der so gefundenen Neuronen gesehen. Die wesentlichen Schritte des Cluster-Algorithmus sind:

1. Bestimme die Mächtigkeit der Voronoimengen V_{ci} aller Neuronen c_i, indem für jedes Muster der Gewinner s_1 gesucht und bei diesem ein Zähler erhöht wird.
2. Entferne tote Neuronen bzw. solche mit einer Trefferzahl $V_{ci}<n$. Obwohl es weniger wahrscheinlich ist, können auch beim GNG-Algorithmus tote Neuronen entstehen. Weiterhin wird ab einer bestimmten Gesamtzahl von Neuronen eine Überrepräsentierung möglich, so kann beispielsweise ein Neuron in der Nähe eines Ausreißers gesetzt werden. Gerade für die Erstellung eines Codebuchs zur Klassifizierung stellt sich die Frage, ob es sinnvoll ist, über Bereiche des Eingaberaumes zu generalisieren, die ganz oder beinahe leer sind. Daher wurden für alle Simulationen Neuronen mit einer Trefferzahl kleiner als Zwei entfernt.
3. Durch induzierte Delaunay-Triangulation werden die Kantenstrukturen gebildet. Entferne zunächst alle Kanten. Suche dann für jeden Datenpunkt die beiden nächsten Gewinner und verbinde sie mit einer Kante, wenn diese nicht bereits existiert. Somit werden Kanten zwischen benachbarten Neuronen nur dann gebildet, wenn die Kante über einen Bereich nichtverschwindender Wahrscheinlichkeitsdichte läuft. Die so entstehende Kantenstruktur ist ein Subgraph der Delaunay-Triangulation. Weiter besteht die Möglichkeit, eine Kante erst dann zu bilden, wenn diese im beschriebenen Algorithmus eine Mindestanzahl von „Kantentreffern“ erhält.
4. Bestimme die mittlere Ausbeute θ_i der Muster im Voronoigebiet jedes Neurons c_i.
5. Betrachte die Neuronen des gefundenen Clusters und die Voronoimengen der zugehörigen Neuronen.

Abbildung 3.60 zeigt ein wachsendes neuronales Gas mit 400 Einheiten in zweidimensionaler Projektion. Die Visualisierung der Daten in dieser Projektion zeigt nicht, ob beispielsweise die Datenpunkte am rechten Bildrand tatsächlich zu einem Cluster gehören. Erst die induzierte Delaunay-Triangulation beweist, dass dies tatsächlich der Fall ist.

In diesem speziellen Cluster liegen 48 Muster mit einer mittleren Ausbeute von 60,78 bei einer Standardabweichung von 4,79. Einzelne Cluster und deren Lage können noch eindeutiger identifiziert werden, indem man einen Cluster optisch markiert und diesen in verschiedenen zweidimensionalen Projektionen betrachtet (siehe Abbildung 3.61)

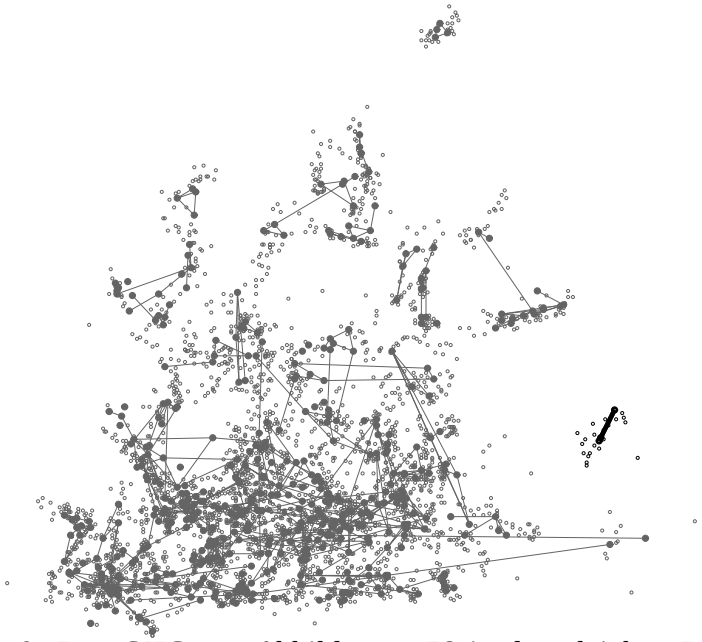

Abbildung 3.60: Das GNG aus Abbildung 3.58 in der gleichen Projektion. Der Cluster am rechten Bildrand bestehend aus zwei Neuronen sowie alle Muster, die in die Voronoigebiete der beiden Neuronen fallen sind markiert. Man sieht hier, dass tatsächlich alle Datenpunkte in der Umgebung dieser drei Neuronen zu einem Cluster gehören. Er besteht aus 23 Mustern mit einer mittleren Ausbeute von 81,5 bei einer Standardabweichung von 3,2.

Abbildung 3.61: Zwei verschiedene Projektionen des GNG aus Abbildung 3.58. Die Neuronen eines Clusters sind als dicke, durchgezogene Linien markiert. Durch Betrachten verschiedener Projektionen können die einzelnen Cluster eindeutig identifiziert werden.

3.3.3 Überwacht wachsendes neuronales Gas (SGNG)

Beim überwacht wachsenden neuronalen Gas wird die Information über die Ausbeute der Muster verwendet, um die Lokalität neuer Neuronen zu bestimmen. Die Algorithmen GNG und SGNG sollten bei komplexen Problemen ähnliche Resultate zeigen, da bei beiden Methoden für jedes präsentierte Muster *(**x**,d)* die Ressourcevariable E_{s_1} des Gewinnerneurons erhöht wird. Einmal um den euklidischen Quantisierungsfehler $|\mathbf{x} - \mathbf{q}_{s_1}|$ im anderen Fall um den Klassifizierungsfehler $|\theta_{s_1} - d|$.
In beiden Fällen werden aber Ressourcen bevorzugt in solchen Gebieten erhöht, wo die Wahrscheinlichkeitsdichte *p(**x**)* im Eingaberaum hoch ist, da die präsentierten Muster zufällig, gemäß der Wahrscheinlichkeitsverteilung gezogen werden. Die lokalen Änderungen der Ressourcevariablen sind auch in ihrer Größenordnung gleich, wie man in Abbildung 3.62 erkennt.
Der gleitende Mittelwert $\Delta\overline{E}$ der Änderung der Ressourcevariablen ist für das SGNG im Prinzip ein Maß für den momentanen Klassifizierungsfehler des Netzes auf den Trainingsdaten, während er für ein normales GNG dem momentanen (euklidischen) Quantisierungsfehler entspricht.

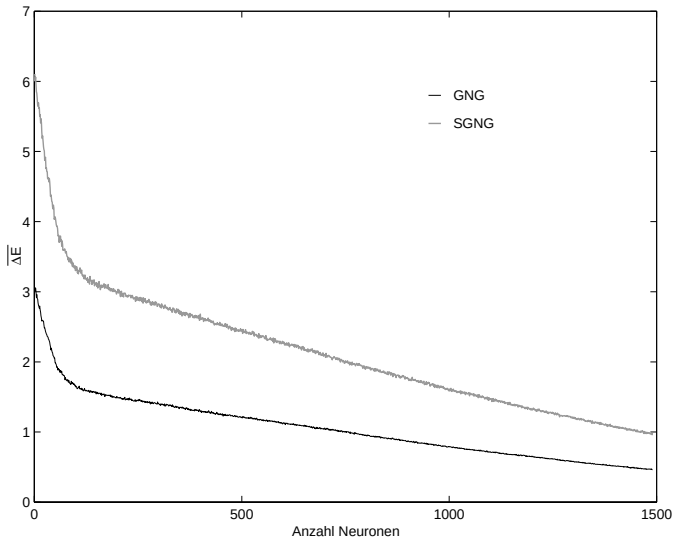

Abbildung 3.62: Vergleich des gleitenden Durchschnitts $\Delta\overline{E}$ für die Änderung der Ressource eines wachsenden neuronalen Gases in überwachter bzw. unüberwachter Form. Die untere schwarze Kurve entspricht dem GNG, die obere graue dem SGNG. Man beachte, dass beide Graphen jeweils verschiedene Größen darstellen. Beim GNG handelt es sich um den euklidischen Quantisierungsfehler, während es für das SGNG ein Maß für die lokale Streuung im Ausbeute-Label ist.

Offensichtlich sind die Topologie des Raums und die gesuchte Abbildung $f : \Re^{22} \to \Re$, die den Eingaberaum auf die jeweilige Ausbeute abbildet, zu komplex, um sie mit der einfachen Erweiterung des SGNG besser zu repräsentieren.

3.3.4 Nächste-Nachbar-Klassifikator

In den nächsten Abschnitten werden verschiedene Klassifizierungsalgorithmen vorgestellt. Um die Generalisierungsfähigkeit der einzelnen Algorithmen zu überprüfen, wurde der Rohdatensatz $(\boldsymbol{x},d)_i$ zufällig in 2500 Trainingsmuster (S_{train}) und 622 Testmuster (S_{test}) zerlegt. Jede Methode wurde mit Hilfe der 2500 Trainingsmuster überwacht trainiert. Dann wurde die Effizienz anhand der Testmuster gemessen, indem für einen trainierten Klassifikator die Ausgabe $y(\boldsymbol{x})$ für alle Testmuster mit der tatsächlichen Ausbeute verglichen wurde. Für die Auswertung ist zu beachten, dass hier wiederum auf die Angabe Prozent bei der Waferausbeute verzichtet wurde.

Ein sehr einfacher statistischer Klassifikator, der nicht auf einem neuronalen Netz basiert, und trotz seines recht einfachen Algorithmus oft erstaunlich gute Ergebnisse erzielt, ist der k-Nächste-Nachbarn-Klassifikator (*k-next-neighbor*, knn), der bereits in Kapitel 3.1 erfolgreich eingesetzt wurde. Dieser sucht einfach für ein zu klassifizierendes Muster $(\boldsymbol{x},d)_T$ die k Trainingsmuster, die ihm euklidisch am nächsten sind, also die Gewinner $s_1, \ldots, s_k$, wenn die 2500 Trainingsmuster als Referenzvektoren ebenso vieler Neuronen betrachtet werden.

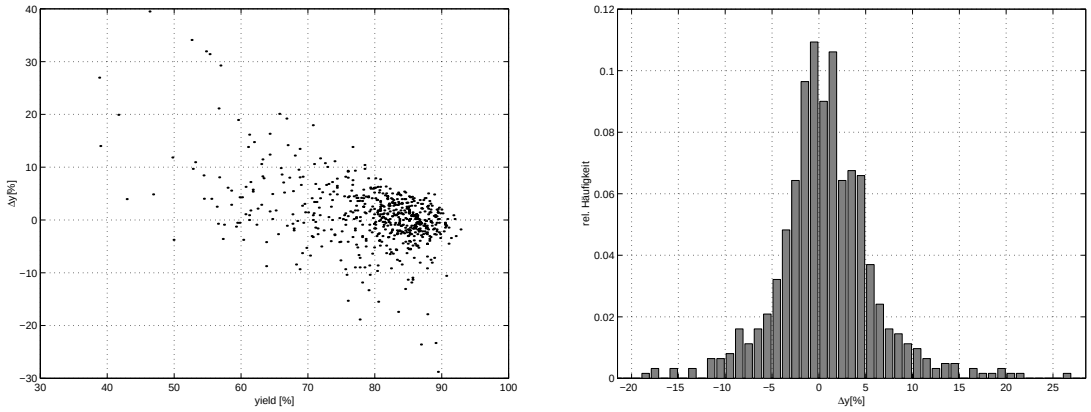

Abbildung 3.63: Fehler des knn-Klassifikators für k=1. *Der mittlere Klassifizierungsfehler liegt bei 4,12. Die Verteilung der Fehler erinnert stark an die Differenz in der Ausbeute nächster Nachbarn in Abbildung 3.55. Der Unterschied liegt darin, dass in Abbildung 3.55 die gesamten 3122 Messdaten verwendet wurden, während für den Klassifikator der Datensatz in 2500 Trainings- und 622 Testmuster zerlegt wurde.*

Die Ausgabe des Klassifikators wird dann als Mittelwert

$$y(\mathbf{x}_T) = \frac{1}{k}\sum_{i=0}^{k} d_{si} \qquad (3.71)$$

angenommen und mit dem tatsächlichen *Yield* d_T des Testmusters verglichen. Abbildung 3.63 zeigt das Ergebnis für k=1. Der mittlere Klassifizierungsfehler Δy liegt bei 4,12, ca. 20 Prozent der Muster werden mit einer Abweichung kleiner 1,0 klassifiziert.

Die folgende Tabelle zeigt den mittleren Klassifizierungsfehler für k>1.

Tabelle 3.14: mittlerer Klassifizierungsfehler für k>1

k	1	2	3	4	5	6	7	8	9	10
Δ_k	4.12	4.26	4.38	4.26	4.47	4.74	4.95	5.58	6.06	6.70

Wie erwartet steigt der Fehler für wachsende k. Allerdings wäre durchaus möglich gewesen, dass der Fehler für kleine k zunächst fällt, als Effekt einer besseren Generalisierung. Jedenfalls stellt dieser Wert von 4,12 einen weiteren Anhaltspunkt für die Beurteilung der folgenden Methoden dar.

3.3.5 Lokal konstante Abbildungen (LCM)

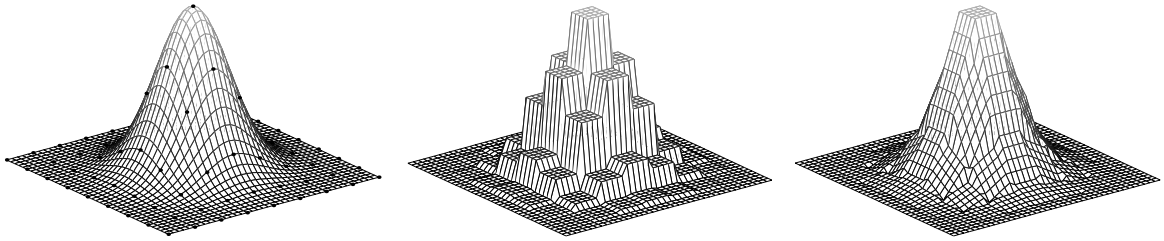

Abbildung 3.64: Darstellung von lokalen Abbildungen anhand der Approximation der Funktion $f(r)=exp(-0,2r^2)$. Stützstellen liegen auf einem quadratischen Gitter bei ganzen Zahlen(x,y), links durch schwarze Punkte angedeutet. In deren Voronoi-Region wird die ursprüngliche Funktion jeweils durch eine konstante bzw. lineare Funktion angenähert.

Unter einer lokal konstanten Abbildung (*local constant mapping*) versteht man allgemein die Näherung einer beliebigen Funktion oder Abbildung $f : \Re^n \to \Re^m$ durch eine Menge an konstanten Abbildungen:

$$f_i : \mathbf{x} \to l_i \qquad \forall \mathbf{x} \in D_i \qquad (3.72)$$

mit

$$l_i = const$$
$$\mathbf{x} \in \Re^m$$
$$D_i \subset \Re^n$$

die jeweils auf einen Unterraum D_i beschränkt sind. Weiterhin gilt,

$$\bigcup_i D_i = \Re^n \qquad (3.73)$$

der Raum wird also vollständig in eine Menge einfach zusammenhängender Unterräume zerlegt, in denen *f* jeweils durch einen konstanten Wert l_i genähert wird. Im allgemeinen Fall wird man für l_i einfach den Mittelwert von *f* in D_i setzen:

$$l_i = \frac{1}{\Omega_i} \int_{D_i} d\mathbf{r} f(\mathbf{r}) \tag{3.74}$$

mit

$$\Omega_i = \int_{D_i} d\mathbf{r}$$

als Volumen des Unterraums. Ist f gegeben durch eine diskrete Menge an Ein-/Ausgabe-paaren $\{(\boldsymbol{x},d)_i\}$ so vereinfacht sich (3.74) zu

$$l_i = \frac{1}{|D_i|} \sum_{j, \mathbf{x}_j \in D_i} d_j \tag{3.75}$$

Dabei bezeichnet $|D_i|$ die Menge an Mustern in D_i. Entscheidend für eine gute Näherung der Abbildung f ist natürlich die Wahl der Unterräume D_i so, dass der Klassenlabel innerhalb dieser Unterräume möglichst homogen ist, also der Fehler bei der Näherung durch den Mittelwert klein bleibt. Genau diese Eigenschaft erfüllen die Voronoigebiete der wachsenden Netze. Es bietet sich also an, für eine durch GNG/SGNG gefundene Verteilung an Referenzvektoren $\{\boldsymbol{q}_i\}$, die Zerlegung $\{D_i\}$ gemäß der Voronoi-Tesselation zu bilden. Dann wird die gesuchte Abbildung durch einen in der jeweiligen Voronoi-Region konstanten Label genähert. Die Referenzvektoren können auf verschiedene Arten gewonnen werden, beispielsweise zufällig aus den Eingabemustern gezogen, per Kohonenkarte, GNG oder SGNG gefunden werden.

Formal sieht der Algorithmus folgendermaßen aus:

1. Belerne eine Kohonenkarte, ein GNG oder ein SGNG mit den 2500 Trainingsmustern S_{train} und finde so Neuronen $\{c_i\}$mit Referenzvektoren $\{\boldsymbol{q}_i\}$ oder wähle diese zufällig aus den Trainingsmustern.
2. Wähle anschließend diejenigen Einheiten aus, deren Referenzvektoren in Gebieten nichtverschwindender Wahrscheinlichkeitsdichte liegen, d.h. solche deren Voronoimenge nicht leer ist.
3. Berechne für alle übrigen Einheiten den Ausbeutelabel als arithmetisches Mittel der Ausbeuten aller Trainingsmuster in ihrer Voronoimenge:

$$l_i = \frac{1}{|V_{c_i}|} \sum_{j, \mathbf{x}_j \in V_{c_i}} d_j$$

4. Klassifiziere dann die Test-Muster, indem jedem der Label des Gewinners s_1 zugeordnet wird, also des Referenzvektors in dessen Voronoigebiet sie fallen:

$$\mathbf{x}_i \rightarrow \mathrm{y}(\mathbf{x}_i)$$
$$\mathrm{y}(\mathbf{x}_i) = l_{s_1}(\mathbf{x})$$

Um die Effizienz des Algorithmus zu überprüfen wurden diese Label mit der tatsächlichen Ausbeute der präsentierten Muster verglichen

$$\Delta d_i = \mathrm{y}(\mathbf{x}_i) - d_i = l_{s_1} - d_i \tag{3.76}$$

und der mittlere Klassifizierungsfehler berechnet:

$$\overline{\Delta d} = E[\Delta d_i] \tag{3.77}$$

Die Ergebnisse für ein GNG der Größe 300 zeigt Abbildung 3.65. Der mittlere Fehler liegt bei 3,49 und damit immerhin schon unter dem in Abschnitt 3.3.1 berechneten Streuwert von 3,82.

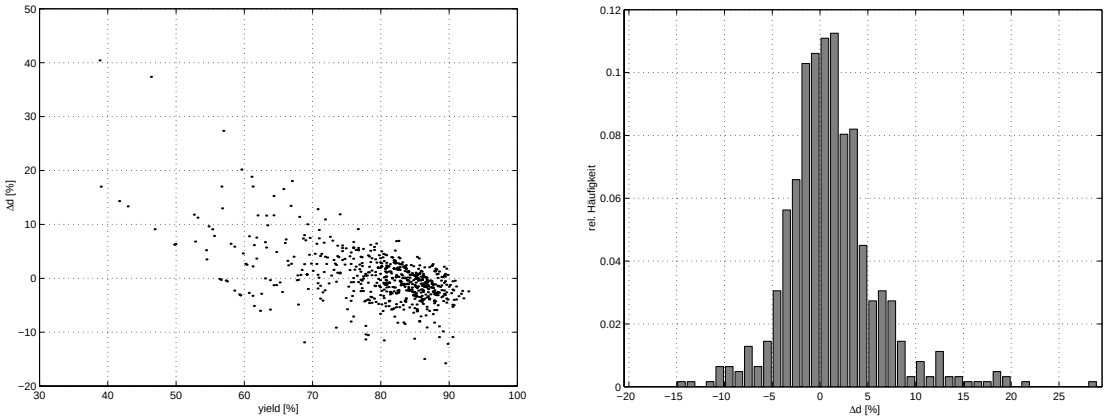

Abbildung 3.65: Klassifikationsfehler von lokal konstanten Abbildungen für 300 Referenzvektoren, die durch ein wachsendes neuronales Gas gefunden wurden. Der mittlere Fehler liegt bei 3,49. Etwa 20% der Muster werden mit einem Fehler kleiner 1,0 klassifiziert, 45% mit einem Fehler kleiner 2,0.

Etwa 20% der Muster wurden auf einen Prozentpunkt oder genauer klassifiziert. Weiterhin sieht man, dass Muster mit hoher Ausbeute wesentlich besser klassifiziert wurden. Die größten Fehler liegen im Bereich kleiner Ausbeuten. Diese Eigenschaft wird von größerer Bedeutung, wenn man bedenkt, dass es im Falle eines tatsächlichen Einsatzes dieses Klassifikators schlimmer wäre, einen Wafer mit hoher Ausbeute unterzubewerten, und daher auszusondern (falsche Zurückweisung), als umgekehrt einen Wafer mit geringer Ausbeute zu hoch einzuschätzen (falsche Akzeptanz).

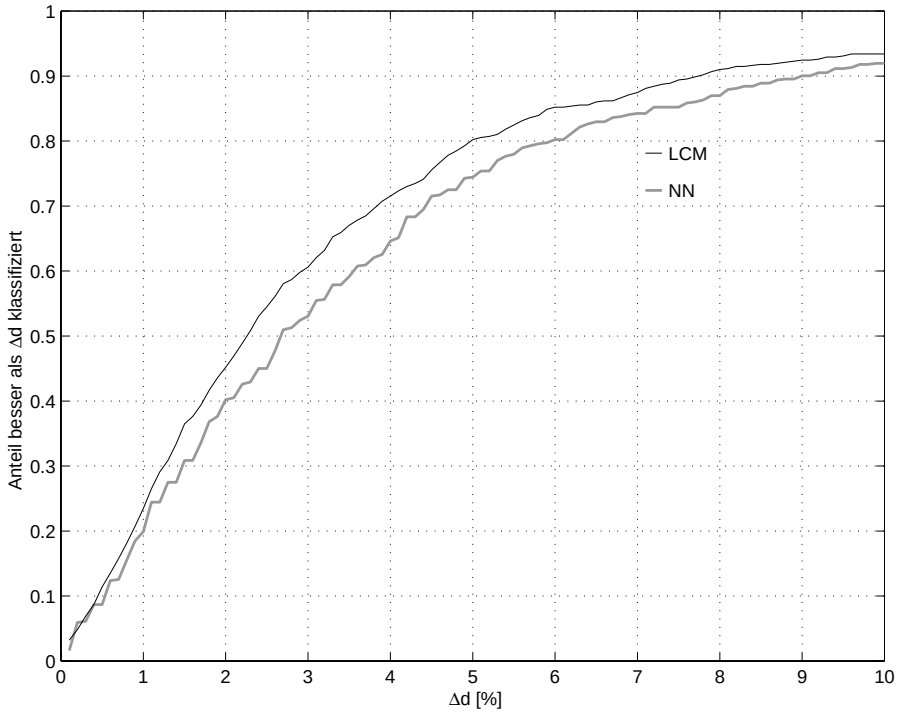

Abbildung 3.66: Vergleich der Klassifizierung des k-next-neighbor-Klassifikators (knn) mit lokal konstanten Abbildungen (LCM) bei 300 Stützstellen. Die Abbildung gibt jeweils an, welcher Anteil der Muster besser als Δd klassifiziert wurde. So wurden beispielsweise für knn 40% der Muster mit einem Fehler kleiner 2,0 klassifiziert, für LCM ca. 45%.

Im folgenden Abschnitt wird eine systematische Versuchsreihe präsentiert, die lokal konstante Abbildungen vergleicht, bei denen die Zentren jeweils durch eine Kohonenkarte, ein GNG, ein SGNG gefunden, oder einfach zufällig aus den Trainingsdaten gezogen wurden. Dabei wurden für jede Methode verschiedene Anzahlen an Zentren gebildet (siehe Tabelle 3.15)

Tabelle 3.15: Zentren-Anzahl für die verschiedenen Methoden GNG bzw. SGNG

GNG/SGNG	20 40 60 ...300 400 500 ...1500
SOFM	49 100 196 400
random	wie GNG/SGNG

GNG bzw. SGNG wurden von 20 bis 300 in Schritten von 20 Einheiten und von 300 bis 1500 in Schritten von 100 Einheiten, die Kohonenkarte in den Größen 7x7, 10x10 und 14x14 trainiert. Für die zufällig gezogenen Zentren wurde die gleiche Anzahl von Neuronen wie für GNG/SGNG verwendet. Um zuverlässige statistische Aussagen zu ermöglichen, wurden für jede Größe und Methode jeweils 20 verschiedene Samples produziert.

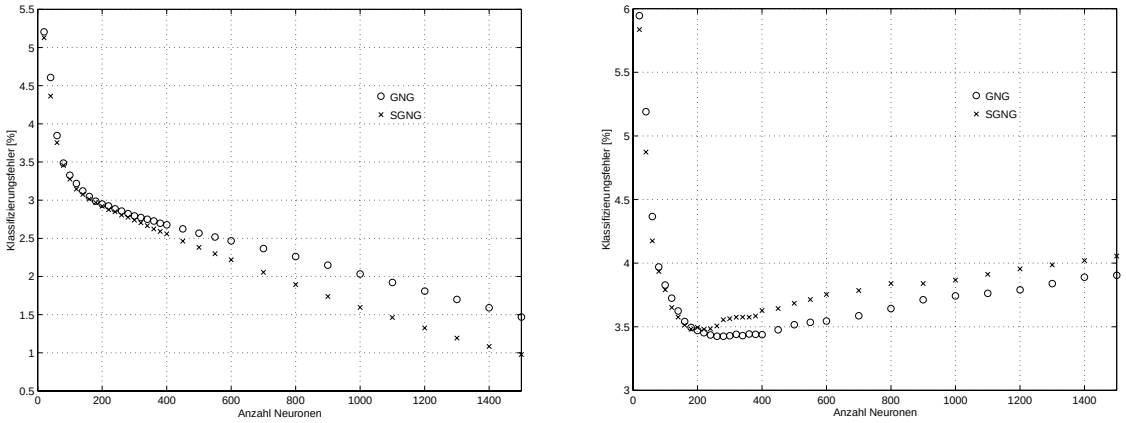

Abbildung 3.67: Vergleich der Klassifizierung der Trainingsdaten (links) bzw. Testdaten (rechts) für verschiedene Netzwerkgrößen durch lokal konstante Abbildungen mit normalen und überwacht wachsenden neuronalen Gasen (GNG/SGNG). Das SGNG lernt die Trainingsdaten schneller zu repräsentieren, zeigt aber gleichzeitig früher Anzeichen von Overfitting. Die besten Ergebnisse werden bei etwa 200 bis 400 Zentren erreicht. Im Mittel liegen dann etwa fünf der 2500 Trainingsmuster im Voronoigebiet jedes Neurons.

Beim Vergleich von GNG und SGNG (siehe Abbildung 3.67) fällt auf, dass das überwachte System ab einer Größe von etwa 300 Einheiten besser auf den Trainingsdaten klassifiziert d.h. es lernt schneller die Topologie des Raums bzw. der zugrundeliegenden Abbildung. Gleichzeitig schneidet es auf den Testdaten ab der selben Größe schlechter ab, und zeigt früher erste Anzeichen von *Overfitting*. In Abbildung 3.68 ist schließlich der Vergleich aller vier Methoden dargestellt.

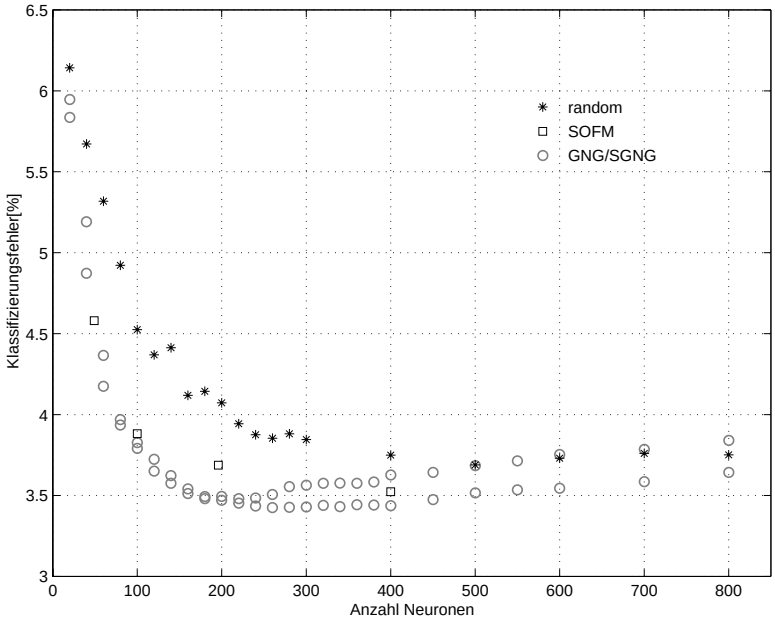

Abbildung 3.68: Vergleich der Klassifizierung von lokal konstanten Abbildungen für verschiedene Zentren, die per Kohonenkarte, GNG, SGNG oder zufälligem Ziehen aus dem Trainingsset gewonnen wurden. Die besten Ergebnisse wurden für wachsende neuronale Gase erreicht. Die Abbildung zeigt auch, dass die Qualität der mit Kohonenkarten erstellten Codebücher besser als erwartet ist.

Wie zu erwarten war, liegen die Werte für zufällig gezogene Zentren jeweils deutlich schlechter als für die wachsenden neuronalen Gase. Für sehr kleine Netze ist kein Unterschied zwischen Kohonenkarten und wachsenden Netzen zu sehen. Beide haben in diesem Bereich den Eingaberaum nur mangelhaft erfasst. Es fällt weiterhin auf, dass die zufällig gezogenen Zentren in Bereichen, wo GNG bereits deutliche Zeichen von *Overfitting* zeigt, mit wachsender Zahl stetig besser werden.
Dies ist damit zu begründen, dass im Gegensatz zur zufälligen Wahl der Zentren, bei SGNG die Klasseninformation mit berücksichtigt wird, und bei GNG die lokale Wahrscheinlichkeits-Dichte. Außerdem wird für die zufällige Wahl der Zentren der Eingaberaum mit wachsender Zahl immer feiner parzelliert, so dass die Verteilung dann auch relativ gut erfasst werden kann.

Als nächstes wird der Vergleich des Klassifizierungsfehlers für normale lokal konstante Abbildungen und solche mit diskreten Klassen durchgeführt. Die weiter unten behandelten RBF-Netze sind auf diskrete Klassen angewiesen. Um einen Vergleichswert zu schaffen, werden die lokal konstanten Abbildungen folgendermaßen modifiziert:

1. Der kontinuierliche Ausbeutelabel d_i wird in 100 diskrete Klassen aufgeteilt, wobei in die jeweilige Klasse die Trainingsmuster mit entsprechender Ausbeute fallen. So werden beispielsweise alle Muster mit einer Ausbeute $40\% \leq d < 41\%$ der Klasse mit dem Index 40 zugeteilt.
2. Bei der Klassifikation der Testmuster wird überprüft, ob die Klassen des Gewinner-Neurons und die des Musters übereinstimmen. Da die Klassen mit der Ausbeute insofern korrelieren, als höhere Indizes auch höhere Ausbeuten bedeuten, ist die Größe $Klasse_{tatsächlich} - Klasse_{output}$ ein Maß für den Klassifizierungsfehler.

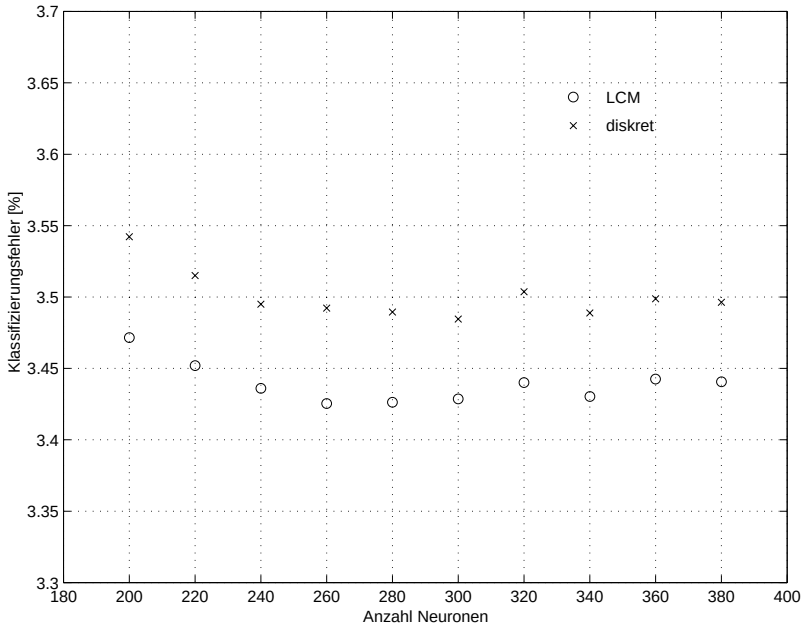

Abbildung 3.69: Vergleich des Klassifizierungsfehlers für normale lokale konstante Abbildungen und solche mit diskreten Klassen. Die Diskretisierung des Klassenlabels liefert nur unwesentlich schlechtere Ergebnisse.

Die Simulationen wurden für die Referenzvektoren von wachsenden neuronalen Netzen der Größen 200 bis 400 durchgeführt. Abbildung 3.68 zeigt den Vergleich der beiden Methoden. Die LCM mit diskreten Klassen liegen jeweils nur um etwa 0,1 schlechter als ihre kontinuierlichen Verwandten.

3.3.6 Wachsende lokal konstante Abbildungen (GLCM)

Bei den überwacht wachsenden neuronalen Gasen ist bereits die Idee verwirklicht, dass neue Einheiten mit dem Ziel eingefügt werden, direkt den lokalen Klassifizierungsfehler auf den Trainingsdaten zu minimieren. Neue Einheiten werden bevorzugt an solchen Stellen des Raums erzeugt, wo die Streuung im Klassenlabel, also der Ausbeute, besonders hoch ist. Beim Gewinnerneuron wird jeweils der Klassifizierungsfehler für ein Muster akkumuliert, die Ressourcevariable E und der Klassenlabel θ des Neurons gemäß

$$\Delta E_{s_1} = \left| \theta_{s_1} - d \right| \tag{3.78}$$

$$\Delta \theta_{s_1} = \eta_\theta \cdot \left(d - \theta_{s_1} \right) \tag{3.79}$$

adaptiert. Obwohl θ aufgrund der stetigen Veränderung des Referenzvektors gewissen Schwankungen unterliegt, entspricht θ zumindest in etwa dem Mittelwert der Ausbeuten in der Voronoi-Region des entsprechenden Neurons. Diesen Umstand machen sich wachsende lokal konstante Abbildungen zunutze. Während des Wachstums können zu beliebigen Zeitpunkten die Testmuster direkt klassifiziert werden, indem jedem Testmuster $(\mathbf{x},d) \in S_{test}$ einfach der Label θ_{s1} des jeweiligen Siegerneurons $s_1(\mathbf{x})$ zugeordnet wird.
Der entscheidende Vorteil dieser Erweiterung der überwacht wachsenden neuronalen Gase ist die mögliche Klassifizierung und Überwachung von nichtstationären Verteilungen. In unserem Beispiel könnte in Form eines kontinuierlich lernenden Systems jeder neu vermessene Wafer zunächst klassifiziert werden, also eine Aussage über die zu erwartende Ausbeute getroffen werden. Nach erfolgter Funktionsmessung können die Gewichte und Label der Siegerneuronen nach dem bekannten Schema aktualisiert werden.

Zum Vergleich wurden auf den Gewichten von 20 wachsenden lokal konstanten Abbildungen nachträglich normale LCMs durchgeführt (siehe Abbildung 3.70). Dabei liegen die wachsenden Systeme nur geringfügig schlechter.

Eine Möglichkeit, den Gesamt-Bias eines Satzes an lokalen Abbildungen zu verringern, ist natürlich die geeignete Wahl der Anzahl an Unterräumen D_i. Andererseits kann man auch versuchen, die Qualität der lokalen Abbildungen selbst zu verbessern, beispielsweise durch Polynome beliebiger Ordnung zu nähern:

$$f_i(\mathbf{x}) = \theta_{s_1} + \mathbf{a}_{s_1}\mathbf{x} + \mathbf{x}\mathbf{B}_{s_1}\mathbf{x} + \ldots \tag{3.80}$$

wobei $\boldsymbol{a}_{s1}$ ein Vektor der Länge *n* ist, $\boldsymbol{B}_{s1}$ eine *n* x *n*-Matrix usw.. Für den linearen Fall (*local linear mapping*, LLM) wird $\boldsymbol{a}_{s1}$ bei jedem Zyklus für ein präsentiertes Muster ($\boldsymbol{x}$,*d*) angepasst gemäß

$$\Delta \mathbf{a}_{s_1} = \eta a \cdot \left(\mathbf{a}_{s_1} - \mathbf{x}\right) \cdot \left(f_{s_1}(\mathbf{x}) - d\right) \tag{3.81}$$

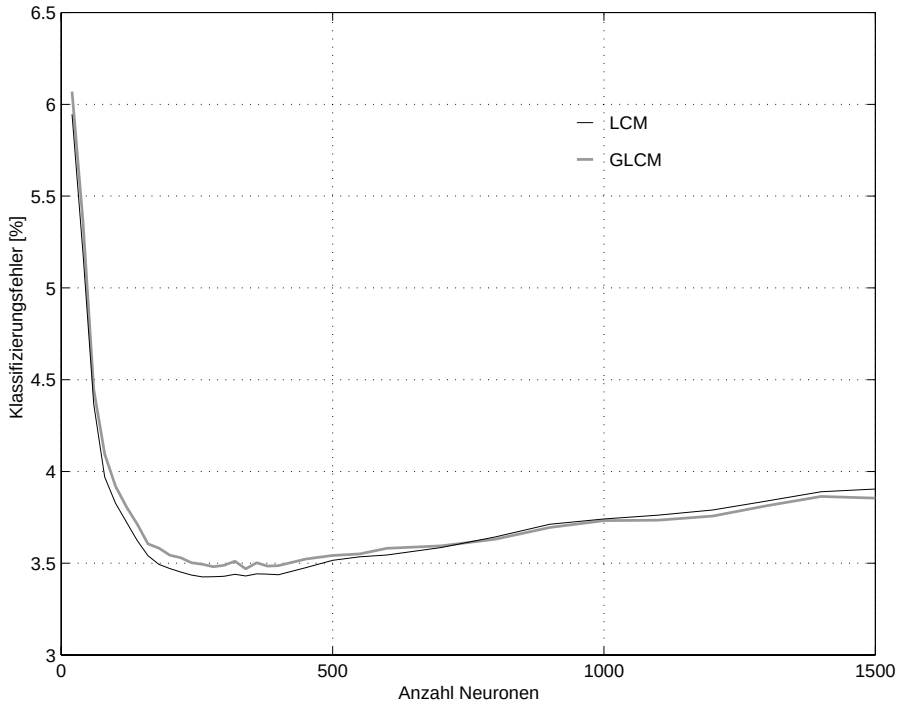

Abbildung 3.70: Vergleich der Klassifizierung von wachsenden lokal konstanten Abbildungen (GLCM) mit lokal konstanten Abbildungen (LCM), die nachträglich auf den gespeicherten Neuronengewichten durchgeführt wurden.

Im Vergleich von linearen mit konstanten wachsenden Abbildungen (siehe Abbildung 3.71) zeigt sich, dass die linearen Varianten schon bei kleinen Netzwerkgrößen besser klassifizieren und früher beginnen, die Trainingsmuster überzurepräsentieren (*Overfitting*).

Allerdings ist die Klassifizierungsleistung insgesamt nicht deutlich besser. Dies könnte durch die Komplexität der genäherten Abbildung zu erklären sein. Eventuell könnten bessere Ergebnisse mit lokalen Abbildungen noch höherer Ordnung, erzielt werden. Allerdings ist der Trainingsaufwand bereits für die linearen Abbildungen beträchtlich. Vergleicht man die Leistung bisher untersuchter Klassifikatoren mit dem Wert der Differenz in der Ausbeute nächster Nachbarn von 3,82 (siehe Abbildung 3.55), so ist wohl bei etwa 3,5 mittlerem Klassifizierungsfehler eine Grenze erreicht, die durch die Struktur und die Verteilung der Daten an sich gesetzt wird und nicht unterschritten werden kann.

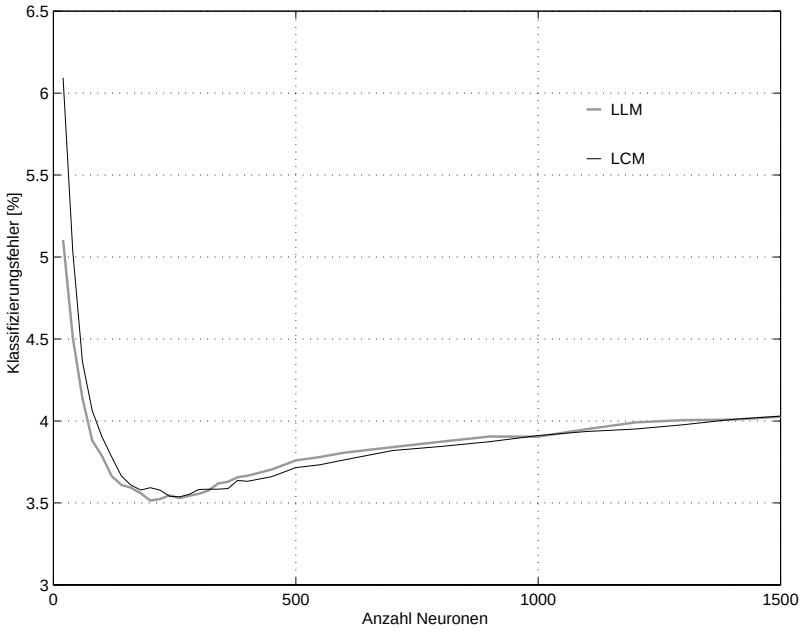

Abbildung 3.71: Vergleich der mittleren Klassifizierung von je 20 lokal konstanten (LCM) bzw. linearen (LLM) Abbildungen. Die linearen Abbildungen können für kleinere Netzwerkgrößen die gesuchte Abbildung besser zu nähern. Im Bereich der besten Ergebnisse ist jedoch kein wesentlicher Unterschied zu sehen.

3.3.7 RBF-Netze

Bei den lokal konstanten Abbildungen liefern wachsende neuronale Gase der Größen 200 bis 380 die besten Ergebnisse. Exakt die gleichen Sätze an Referenzvektoren wurden daher für das Training von RBF-Netzen verwendet. Es kamen also 10 x 20 = 200 verschiedene Verteilungen der Zentren zum Einsatz (je 20 der Größen 200 bis 380). Für jeden dieser 200 Sätze wurden zehn RBF-Netze trainiert, also insgesamt 2000. Die Muster wurden wieder je nach Ausbeute in diskrete Klassen von 1 Ausbeuteprozent Breite eingeteilt. Tatsächlich wurden nur 60 Klassen mit Wafern von 35% bis 95% Ausbeute verwendet, da keine Wafer außerhalb dieser Grenzen lagen.
Bei jedem RBF-Netz werden 75000 Lernzyklen durchgeführt und die Ausgabe-Schicht nach der Delta-Regel belernt, wobei die Lernrate exponentiell von 0,3 auf 0,01 abfällt. Abbildung 3,72 vergleicht den mittleren Klassifizierungsfehler der RBF-Netze mit lokal konstanten Abbildungen auf den selben Zentren.

Die Fehlerbalken für die lokal konstanten Abbildungen wären in dieser Darstellung nicht sichtbar. Die RBF-Netze schneiden für alle Netzwerkgrößen um etwa 1,5 schlechter ab. Dieser Wert ist nur minimal durch die Wahl der Trainingsparameter beeinflussbar. Diese doch signifikante Differenz ist nicht allein durch die künstliche und willkürliche

Klasseneinteilung zu erklären. Schließlich klassifizieren lokal konstante Abbildungen mit diskreten Klassen nicht wesentlich schlechter als ihre kontinuierlichen Verwandten.
Die Komplexität der gesuchten Abbildung ist natürlich immens. Schließlich versucht man, betrachtet man nur die Ausgabe-Schicht, eine Abbildung von einem 200- bis 380-dimensionalen in einen etwa 60-dimensionalen Raum zu finden. Es ist durchaus denkbar, dass für das Training der Perzeptronschicht des RBF-Netzes, die folglich aus etwa 1200 bis 2000 Gewichten besteht, wesentlich mehr Lernzyklen, oder andere Methoden, wie z.B. die der Pseudoinversen, erforderlich wären. Untersuchungen bis zu 500000 Zyklen bei verschiedenen Abkühlschemen und Lernraten zeigten jedoch keine Verbesserung des Klassifikationsfehlers.

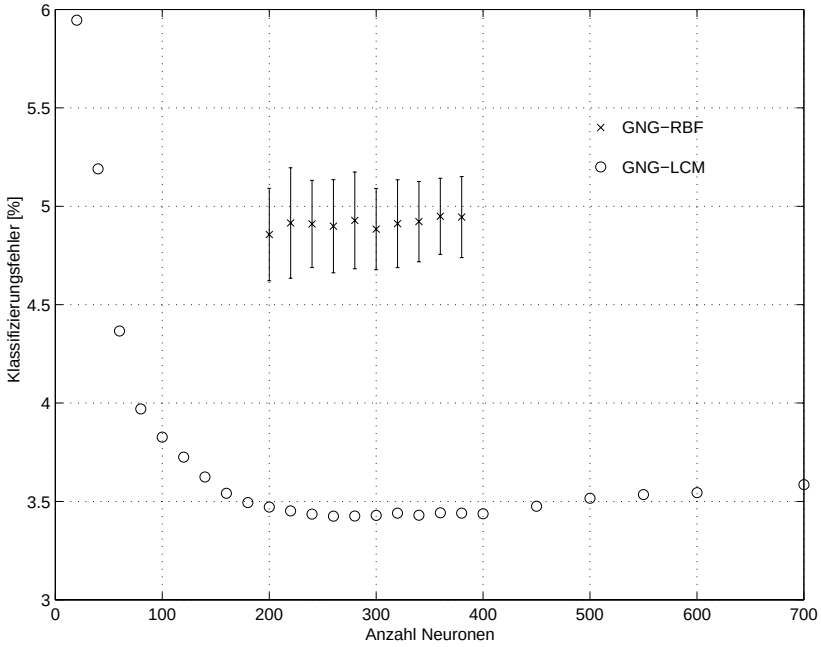

Abbildung 3.72: Vergleich des Klassifizierungsfehlers von lokal konstanten Abbildungen mit RBF-Netzen, die jeweils mit den selben Zentren trainiert wurden. Die RBF-Netze klassifizieren für alle Netzwerkgrößen um etwa 1,5 Prozentpunkte schlechter.

Da für das Training der Gewichte jedes Perzeptrons eine Sollaktivität gleich 1/0 (präsentiertes Muster entspricht/entspricht nicht der dem Perzeptron zugeordneten Klasse) vorgegeben wird, können im Klassifizierungs-Prozess die Aktivitäten der Ausgabeschicht direkt als *a-posteriori* Wahrscheinlichkeiten interpretiert werden. Die Methode der RBF-Netze könnte damit um einen Akzeptanz-Schwellenwert erweitert werden. Ein Muster würde dann nur als klassifiziert gelten, wenn die Aktivität des stärksten Ausgabeneurons ein gewisses Minimum überschreitet. Andernfalls wird die Zuordnung zu einer Klasse verweigert. So würden beispielsweise extreme Ausreißer, die keines der Neuronen der verdeckten Schicht wesentlich aktivieren, von vornherein zurückgewiesen. Entsprechende Versuche führten aber nicht zu befriedigenden Ergebnissen.

Weiterhin wurde auch versucht eine Art „Fuzzy“-Arithmetik auf der Basis der Wahrscheinlichkeitsinterpretation der Perzeptronaktivitäten in den Klassifizierungsprozess einzubringen. Dazu wurde der Vektor $\boldsymbol{y}$ der Ausgabe-Aktivitäten auf Spur 1 normiert und

dann das Skalarprodukt mit den entsprechenden Klassenzugehörigkeiten gebildet. Doch auch dieser Versuch führte nicht zu besseren Ergebnissen.

Um zu untersuchen, ob die simulierten RBF-Netze überhaupt in der Lage sind, ein Klassifikationsproblem zufriedenstellend zu lösen, wurde ein Standardproblem untersucht. Im *vowel-recognition*-Problem, aus der Sammlung der *CMU-Benchmarks*, handelt es sich um zehndimensionale Daten, die aus der Aussprache von Vokalen in der amerikanischen Sprache gewonnen wurden. Der Datensatz unterteilt sich in 528 Trainings- und 462 Testdatensätze und wurde detailliert von Robinson [Robinson89] untersucht.
Er untersuchte verschiedene Modelle, unter anderem Singlelayer- und Multilayer-Perzeptrons, sowie klassische RBF-Netze nach Moody und Darken [Moody89]
Das beste Ergebnis in dieser Arbeit zeigte der *k-next-neighbor*-Klassifikator mit 260 (56%) korrekt zugeordneten Mustern.

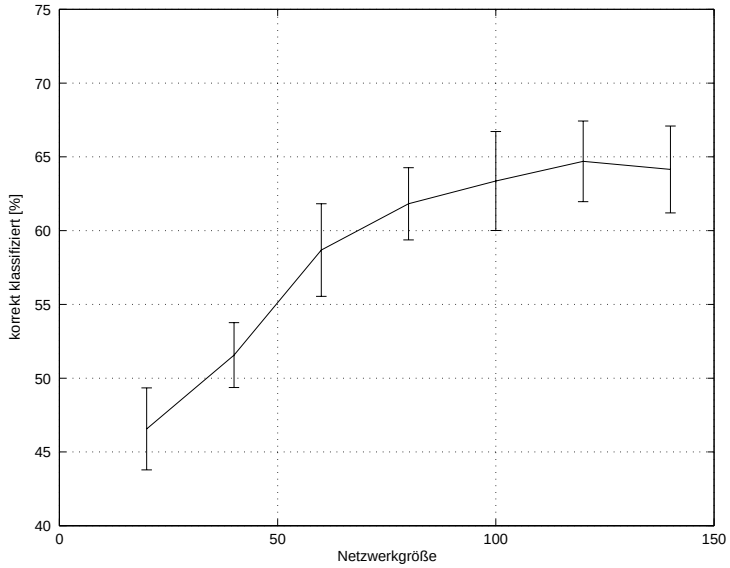

Abbildung 3.73: Korrekte Klassifizierung der Testdaten des vowel-recognition-Problems durch RBF-Netze. Die Referenzvektoren werden von wachsenden neuronalen Netzen der Größen 20 bis 140 bestimmt. Bei einer Größe von 120 Einheiten beträgt der Anteil korrekt klassifizierter Muster 65%. Dieses Ergebnis liegt deutlich über dem für klassische RBF-Netze bzw. Multilayer-Perzeptrons [Robinson89]

Auf diesen Testmustern wurden 20 GNGs trainiert, die in Schritten von 20 Einheiten bis zu einer Maximalgröße von 140 wuchsen. Auf diesen Zentren wurden, unter Verwendung der selben Parameter wie oben, RBF-Netze trainiert. Die besten Ergebnisse (siehe Abbildung 3.73) zeigten Netze der Größe 120, die im Mittel 301 Muster (64%) korrekt klassifizierten. Wobei maximal 320 (69%) Muster korrekt zugeordnet wurden. Diese Werte liegen deutlich über den von Robinson erzielten Ergebnissen. Somit erreicht die Kombination von wachsenden Strukturen mit RBF-Netzen sehr gute Ergebnisse für dieses Klassifikationsproblem.

3.3.8 Diskussion

Ziel dieses Kapitels war, verschiedene neuronale Netzwerke zur Mustererkennung und Klassifizierung hochdimensionale Datensätze aus der Produktion von Silizium-Wafern zu untersuchen.
Beim verwendeten Datenmaterial handelt es sich um die PCM-Parameter und die zugehörige Ausbeute der einzelnen Wafer. Pro Wafer stehen ca. 65 Parameter zur Verfügung, wobei in einer Vorselektion nur die wichtigsten 22 Parameter ausgewählt und anschließend näher untersucht wurden. Diese sogenannten PCM-Parameter (*process control monitoring*) werden an eigens aufgebrachten Strukturen, die in den Zwischenräumen der eigentlichen Chips erzeugt werden, gemessen. Der fertig prozessierte Wafer wird nach erfolgter PCM-Messung zur Funktionsmessung zugelassen, falls die PCM-Parameter keine besonders auffälligen Werte zeigen. Bei der Funktionsmessung wird die Ausbeute, d.h. der Anteil an funktionstüchtigen Bauteilen, ermittelt.

Da die PCM-Strukturen die gleichen Prozessierungsschritte durchlaufen wie die eigentlichen funktionalen Strukturen des Wafers, sollte ein direkter Zusammenhang zwischen den PCM-Messergebnissen und dem Funktionstest bestehen. Die eigentliche Aufgabenstellung lag also darin, eine Korrelation zwischen den PCM-Parametern und der Ausbeute zu beschreiben, also eine Abbildung, repräsentiert durch eine diskrete Menge an Paaren, zu approximieren.

Nach der Normierung der Daten wurde in der Vorverarbeitung die Voraussetzung für einen erfolgreichen Einsatz eines Klassifikationsalgorithmus untersucht.
Die Verteilung der euklidischen Abstände der PCM-Messwerte zum jeweils nächsten Nachbarn im Eingaberaum zeigt, dass die überwiegende Mehrzahl der Daten ähnliche, im Sinne von geringer euklidischer Distanz, Eingabemuster aufweisen. Ausreißer existieren sowohl bzgl. der Ausbeute als auch bzgl. des Abstandes zum nächsten Nachbarn, wobei diese Abstände nicht korrelieren. Es sind sowohl Bereiche im Phasenraum mit relativ großer Dichte vorhanden, bei denen die Ausbeute gering ist, als auch Singularitäten mit großer Ausbeute.
Damit kann auch für diese Daten die Annahme, die bereits in Kapitel 3.1 getroffen wurde, bestätigt werden. Bereiche großer Phasenraumdichte stellen normale Prozessierungen mit guter Ausbeute dar. Der Umkehrschluss gilt dabei nicht, d.h. ein Bereich geringer Phasenraumdichte stellt nicht unbedingt eine schlechte Prozessierung, aber auf jeden Fall eine abnorme Prozessierung, dar.

Die Darstellung der Ausbeutedifferenz zum nächsten Nachbarn gegen die Ausbeute zeigt wiederum einen Bereich hoher Phasenraumdichte der normalen, guten Prozessierungen. Interessant ist allerdings, dass auch hier Ausreißer vorhanden sind, d.h. benachbarte Wafer haben eine wesentlich höhere oder wesentlich geringere Ausbeute, obwohl die euklidische Distanz gering ist. Dies deutet darauf hin, dass benachbarte Bereiche im Phasenraum in unterschiedliche Klassen fallen und liegt, falls es sich hier nicht um systematische Messfehler handelt, darin begründet, dass die PCM-Parameter unterschiedlich sensitiv bzgl. der Ausbeute sind. Im statistischen Mittel weichen die Ausbeuten benachbarter Wafer um 3,82

Ausbeuteprozent voneinander ab und bilden damit eine Art natürliche Referenz für generalisierende Klassifikatoren.

Wachsende neuronale Gase (*Growing Neural Gas*, GNG) bilden eine topologie-erhaltende Karte des Eingaberaumes mit beliebiger Genauigkeit [Bruske95]. Durch ihre Eigenschaft, neue Neuronen in den Bereichen maximaler Ressourcen einzufügen, entstehen neue Einheiten nur in Bereichen des Phasenraumes, wo die Wahrscheinlichkeitsdichte endlich ist. Die Kanten zwischen zwei Neuronen werden gebildet, wenn sie ähnliche Referenzvektoren haben. Durch die Verwendung einer Altersvariablen wird erreicht, dass keine toten Neuronen entstehen können und Kanten nur in den Bereichen gebildet werden, wo die Wahrscheinlichkeitsdichte ungleich Null ist. Damit kann dieser Algorithmus auch zum Auffinden von Clustern in hochdimensionalen Datenräumen verwendet werden.
Durch die Projektion des 22-dimensionalen Datenraumes auf zwei Dimensionen zeigen sich Bereiche, die vorhandene Cluster vermuten lassen. Durch die induzierte Delaunay-Triangulation wird gezeigt, dass es sich tatsächlich um Cluster handelt. Untersuchte Cluster zeigen eine Homogenität bzgl. der Ausbeute und eine auf das verwendete euklidische Distanzmaß bezogene, Ähnlichkeit. Für die untersuchten PCM-Daten der Si-Wafer ergeben sich damit Bereiche im Phasenraum mit ähnlichen Eigenschaften und zugehörigen Referenzvektoren. Diese können für den Fall niedriger Ausbeuten auch als typische Ausfallmuster definiert werden. GNG-Netze sind dabei in der Lage, entsprechende Muster bzw. Cluster zu finden. Ein Vergleich mit Kohonenkarten zeigt, dass der quadratische Fehler für GNG, bei gleicher Anzahl verwendeter Neuronen, geringer ist.

Die wachsenden neuronalen Gase erzeugen ein unbewertetes Codebuch und finden Repräsentanten im Eingaberaum so, dass die Wahrscheinlichkeitsdichte der Muster im Eingaberaum gut approximiert wird. Will man allerdings mit diesem System Referenzvektoren zur Klassifizierung bestimmen, z. B. die Zentren für RBF-Netze oder Stützpunkte für lokal konstante Abbildungen, ist dieser Ansatz prinzipiell wenig sinnvoll. Ein Ansatz, diese Schwächen auszugleichen und die Klassenlabel mit in den Trainingsprozess einzubeziehen, wurde von Fritzke [Fritzke94] diskutiert und von Leim [Leim00] um die kontinuierliche Darstellung der Klassenlabel erweitert.
Vorteil dieses Verfahrens ist, dass bei jedem Lernschritt beim Gewinnerneuron eine Größe für die lokale Varianz der Klassenlabel berücksichtigt wird und neue Neuronen in solchen Gebieten eingefügt werden, wo die Streuung der Klasseneinteilung im Eingaberaum hoch ist.
Ist eine gute Separation des Eingaberaumes bzgl. der Klasseneinteilung gegeben, erwartet man für beide Algorithmen gleiche Ergebnisse. Für den Fall von komplexen Datenräumen mit ähnlichen Eingabemustern aber völlig unterschiedlicher Klasseneinteilung wird der SGNG Algorithmus eine bessere Clustereinteilung leisten können.
Für die verwendeten Testdatensätze ergeben sich allerdings keine wesentlichen Änderungen bzgl. des Lernfehlers. Dies kann daran liegen, dass die gesuchte Abbildung des Eingaberaumes auf die Ausbeute zu komplex ist, um sie mit dieser, relativ einfachen, Erweiterung für SGNG zu verbessern. Da die Ergebnisse bei GNG bereits eine Homogenität bzgl. der Ausbeute in den gefundenen Clustern aufweisen, ist dieses Ergebnis jedoch als

weiteres Indiz dafür zu sehen, dass die Separation des Eingaberaumes, und Erstellung eines Codebuches, bereits durch GNG zufriedenstellend gelöst wurde.

Im wesentlichen wurden drei Arten von Klassifikatoren näher untersucht. Die Rohdatensätze wurden in 2500 Trainingsmuster und 622 Testmuster unterteilt. Ein sehr einfacher statistischer Klassifikator, der nicht auf einem neuronalen Netz basiert und trotz seines recht einfachen Aufbaus oft erstaunlich gute Ergebnisse erzielt, ist der *k-next-neigbor*-Algorithmus (knn).
Der knn-Algorithmus erreicht einen Klassifizierungsfehler von 4,12 Ausbeuteprozent für k=1 und bestätigt damit die Schlussfolgerungen bei GNG und SGNG, dass für die überwiegende Mehrheit der Testdaten ähnliche Bereiche des Eingaberaumes ähnliche Ausbeuten zur Folge haben. Für wachsende k steigt der Klassifizierungsfehler. Dies ist klar, werden doch unähnlichere Eingabemuster zur Klassifizierung herangezogen. Es wäre allerdings auch möglich gewesen, dass der Fehler für kleine k, als Folge einer besseren Generalisierung, zunächst fällt. Diese Vermutung hat sich nicht bestätigt.

Lokal konstante Abbildungen zerlegen den Phasenraum in n zusammenhängende Unterräume in denen der zugehörige Ausgabewert durch eine skalare Größe, üblicherweise den arithmetischen Mittelwert, genähert wird. Entscheidend für eine gute Abbildung ist die Wahl der Unterräume. Die Klasseneinteilung innerhalb der Unterräume sollte dabei möglichst homogen sein. Diese Eigenschaft erfüllen die Voronoigebiete der wachsenden Netze und werden deshalb, neben der Bestimmung durch Kohonenkarten und einfacher zufälliger Auswahl von Testmustern, zur Bestimmung des Codebuches herangezogen und verglichen.
Die besten Ergebnisse wurden dabei für die wachsenden neuronalen Gase erreicht. Das SGNG lernt dabei schneller, zeigt aber auch eher Zeichen von *Overfitting*. Die besten Ergebnisse wurden für etwa 200 – 400 Neuronen erreicht. Im Mittel liegen dann fünf Trainingsmuster im Voronoigebiet jedes Neurons. Die Qualität der mit Kohonen erstellten Codebücher ist dabei besser als angenommen. Mit einem mittleren Klassifizierungsfehler von 3,4 Ausbeuteprozent erreichten damit die überwachten neuronalen Gase mit lokal konstanten Abbildungen bei einer Größe von 200 – 400 Neuronen die besten Ergebnisse. Dieser Wert liegt deutlich unter der Differenz nächster Nachbarn von 3,81 Ausbeuteprozent.

Bei den überwachten wachsenden neuronalen Gasen werden neue Neuronen mit dem Ziel eingefügt, den lokalen Klassifizierungsfehler zu minimieren. Das bedeutet, dass neue Referenzvektoren dort eingefügt werden, wo die Streuung der Klassenzugehörigkeit groß ist. Den Klassenlabel der Neuronen entsprechen dabei in etwa der mittleren Ausbeute der zugehörigen Voronoigebiete. Die wachsenden, lokal konstante Abbildungen (GLCM) nutzen diese, bereits während des Trainingsvorganges vorhandene, Klasseninformation. Während des Wachstums können zu beliebigen Zeitpunkten Testmuster direkt klassifiziert werden, indem jedem Testmuster einfach der Klassenlabel des jeweiligen Siegerneurons zugeordnet wird.

Der entscheidende Vorteil dieser Erweiterung von überwachten wachsenden neuronalen Gasen ist die mögliche Klassifizierung und Überwachung von nichtstationären Verteilungen. Für die Daten der prozessierten Si-Wafer bedeutet das, dass jeder neu gemessene Wafer

zunächst klassifiziert, also eine Aussage über die zu erwartende Ausbeute getroffen wird. Nach erfolgter Funktionsmessung können die Gewichte und Label der Siegerneuronen nach den bekannten Schemata aktualisiert werden. Diese Eigenschaft ist nicht nur für die in diesem Kapitel untersuchten PCM-Messungen interessant. Auch bei den in Kapitel 3.1 und 3.2 untersuchten Zeitreihen zeigen sich Nichtstationaritäten der Signalverläufe. Polymerablagerungen auf den Fenstern der Reaktorkammern bewirken, das durch steigende Zahl von Prozessierungen die Intensität spektroskopischer Signale abnimmt. Bei einer Kammerreinigung werden diese Ablagerungen wieder entfernt. Für diese Art von Signalverläufen kann durch GLCM-Verfahren der nichtstationäre Charakter der zugrundeliegenden Systemdynamik berücksichtigt werden.

Im Vergleich von RBF-Netzen mit LCM-Verfahren weisen die RBF-Netze für alle Netzwerkgrößen eine um 1,5 Ausbeuteprozent schlechtere Klassifikationsleistung auf. Dieses etwas unverständliche Ergebnis kommt vermutlich daher, dass für die RBF-Netze die effektive Entscheidungsregion eine beliebige Linearkombination von Hypersphären darstellt. Diese Radien der Sphären sind willkürlich gewählt und nur mit einem stark konvergierten Ausgabeperzeptron können gute Ergebnisse erwartet werden. Für das Training dieser Schicht müssten deshalb evtl. wesentlich mehr Lernzyklen aufgewendet werden, was allerdings in der Praxis nicht anwendbar ist.

Zum Nachweis, dass diese Form von RBF-Netzen in der Lage sind, typische Klassifikationsprobleme zufriedenstellend zu lösen, wurde das *vowel-recognition*-Problem aus der Serie der *CMU-Benchmarks* untersucht. Die Zentren der verdeckten Schicht wurden durch wachsende neuronale Gase ermittelt. Für diese Aufgabenstellung lag der Anteil korrekt klassifizierter Muster mit etwa 65% deutlich über in der Literatur berichteten Ergebnissen und zeigt, dass die Klassifizierung von PCM-Daten eine Herausforderung an Klassifikatoren darstellt.

Insgesamt scheint bei einem Klassifikationsfehler von etwa 3,4 Ausbeuteprozent eine Grenze erreicht zu sein, die durch die Struktur der Daten vorgegeben ist und nicht unterschritten werden kann. Für einen tatsächlichen Einsatz der vorgestellten Methoden ist vor allem wichtig, dass die größeren Klassifikationsfehler stets im Bereich geringer Ausbeuten gemacht wurden. So würde häufiger ein Wafer mit geringer Ausbeute zu hoch eingeschätzt (falsche Akzeptanz) als dass ein Wafer mit hoher Ausbeute fälschlicherweise vorzeitig aussortiert würde (falsche Zurückweisung).

4. Zusammenfassung

Der Produktionsprozess integrierter Halbleiterbausteine ist ein komplexer Fertigungsablauf mit mehreren hundert Einzelprozessschritten. Auf Si-Wafern (monokristalline Silizium-Scheiben mit 150, 200 und 300mm Durchmesser und ca. 0,5 mm Dicke) werden in einem rechteckigen Muster mehrere hundert Bauteile wie Mikroprozessoren, Speicher oder spezielle Bauteile für Mobilfunk und Automobilindustrie gefertigt.
Während der Prozessierung werden an einigen Anlagen bis zu 70 Prozessparameter (Temperatur, Druck etc.), mit einer Samplingrate von ca. 1 Hz, aufgezeichnet. Nach wichtigen Prozessschritten werden sogenannte *Inline*-Messungen der Prozessergebnisse, wie Schichtdicken aufgetragener Schichten, Linienbreiten etc. durchgeführt und mit SPC (*Statistical Process Control*) [Dietrich95] überwacht. Nach der Prozessierung der Scheiben im sogenannten *Frontend* werden pro Wafer an zu diesem Zweck auf der Scheibe aufgetragenen Teststrukturen die Einsatzspannungen, Ströme und andere Parameter gemessen und pro Wafer gemittelt. Es stehen ca. 65 dieser PCM-Werte pro Wafer zur Verfügung. Der abschließende Funktionstest der einzelnen Bauteile auf der Scheibe ist sehr aufwendig und teuer. Er wird deshalb nur durchgeführt, wenn die PCM-Werte innerhalb vorgeschriebener Toleranzen liegen. Fehlerhafte Bauteile werden markiert und nach dem Zersägen der Wafer weggeworfen. Im *Backend* werden die Bauteile dann in die Gehäuse montiert und verdrahtet.

Aufgrund der Komplexität des Einzel- bzw. des Gesamtprozesses und aufgrund der großen Datenmengen sind neuronale Netze Gegenstand intensiver Untersuchungen in der Halbleiterindustrie. Hauptanwendungen sind die Modellierung von Einzelprozessen und Prozesssequenzen zur Optimierung der Rezepteinstellungen, die Anlagenregelung in Echtzeit und die Fehlererkennung, Klassifikation und Diagnose von Anlagen- und Prozessproblemen.

Neuronale Netze in der Halbleiterfertigung

Wang et al. [Wang95] modellierten und optimierten z.B. *Recipe*-Einstellungen eines CVD-Abscheideprozesses (*Chemical Vapor Deposition*) um optimale Schichtdicken und Uniformitäten zu erreichen. Ghazanfarian et al. [Ghaz97] verwenden ein einfaches dreischichtiges neuronales Netzwerk, um *Misalignment* von Belichtungsanlagen zu minimieren. Falsches Alignment in der Lithographie führt dazu, dass die einzelnen Schichten des Halbleiterbauteils nicht exakt übereinander liegen und damit z.B. Kontaktierungen zwischen den einzelnen Schichten fehlschlagen.

Zur Modellierung eines Plasmaätzprozesses verwenden Salam et al. [Salam97] ein Backpropagation-Netzwerk und als Eingabe die eingestrahlte Leistung, den Druck in der Reaktorkammer und die Kammergröße. Als Netzwerkausgabe verwendeten die Autoren die Ionenanzahl, absorbierte Leistung und die elektrische Feldstärke in der Kammer und verwenden dieses Modell als Grundlage echtzeitfähiger Regelstrategien. Card [Card00] stellt einen universellen Controller für Halbleiter Fertigungsprozesse vor und zeigt exemplarisch für den Trockenätzbereich dessen Anwendbarkeit. Eingabewerte stellen ca. 40 Prozessparameter dar und modelliert werden die Ätzrate, deren Selektivität zwischen Oxid und Fotolack, und die Standardabweichung der Ätzrate. Weitere Beispiele für die Modellierung von Trockenätzprozessen siehe [Bushmann97], [Lim99].

Unter dem Begriff *Run-to-Run Control* wird seit kurzer Zeit versucht systematische Effekte der Anlagen, des Eingangsmaterials und der Vorprozesse für die Bestimmung der Einzelprozessrezepte zu berücksichtigen um eine Stabilisierung der Prozessergebnisse zu erreichen. Wang et al. [Wang96] zeigen, dass auch für die Modellierung von Einzelprozesssequenzen und Optimierung der Einzelprozessrezeptparameter unter Berücksichtigung der Vorprozesse neuronale Netze geeignet sind.

Das zweite Anwendungsgebiet neuronaler Netze, und Gegenstand der Untersuchungen dieser Arbeit, ist die Klassifikation, Fehlererkennung, Diagnose und Detektion typischer Ausfallmuster, die auf bestimmte Anlagenprobleme, bzw. fehlerhafte Prozesse schließen lassen. Verschiedene Netzwerkansätze wurden für *Yield*-Daten [Barrios98], [Rietman01] und Defektdichte-Daten [Chen00], *Inline*-Messdaten für Schichtdicken nach einem CVD- Prozess [Bhatikar99] und für Prozessparameter (z.B. für Trockenätzen [Wise99], PECVD [Chen98]) untersucht.

Chen et al. [Chen00] untersuchen räumliche Verteilungen der Ausbeuten um fehlerhafte Prozessierungen zu klassifizieren und eine Prognose der Problemursache zu geben.
In ihrer Arbeit verwenden die Autoren Art1- und SOM-Netzwerke und können verschiedene Muster wie Ringe verschiedener Größe und Kratzer klassifizieren, für Testdaten daraufhin potentielle Problemursachen und neue, bisher noch nicht aufgetretene Probleme detektieren. Die Analyse dieser Daten hat allerdings den entscheidenden Nachteil, dass auf Probleme relativ spät reagiert und die Problemursache erst nach einer relativ langen Zeitspanne detektiert wird.

Chen et al. [Chen98] schlagen verschiedene Methoden zur Überwachung von Prozessanlagen der Halbleiterproduktion, basierend auf der Analyse multisensorischen Zeitreihen *in situ* aufgezeichneten Prozessparametern an PECVD-Equipments (*Plasma Enhanced Chemical Vapor Deposition*) einer Silicon Foundry in Taiwan vor. Sie benutzen PCA um Gruppierungen von Prozessparametern mit hohen Kreuzkorrelationen zu finden. Die Wichtigkeit der einzelnen Parametergruppen wird aufgrund der in diesen Parametern vorhanden Varianz beschrieben, durch Prozessverantwortliche überprüft und eine Gruppierung in „kritische“, „wichtige“ und „nebensächliche“ Parameter durchgeführt.

Für jeden Parameter wird ein Zielwertwert, eine obere und untere Spezifikationsgrenze definiert und überwacht. Einfache Grenzwertprüfungen können damit schwerwiegende Anlagenprobleme detektieren. Zur Zustandsüberwachung der Anlagen wird ein „System-Health-Index“ als Produkt der gewichteten Summen der PCA-Scores mit den jeweiligen Koeffizienten für deren Wichtigkeit gebildet. Dieser „Health-Index“ verschafft damit einen Überblick über den derzeitigen Anlagenzustand in Abhängigkeit von allen Prozessvariablen.
Üblicherweise werden zur Produkt- und Anlagenüberwachung nach der Prozessierung von produktiven bzw. Testscheiben skalare Messwerte, wie Schichtdicken, Linienbreiten, Partikel etc. aufgenommen und mit SPC-Methoden (*Statistical Process Control*) [Shewhart31], [Montgomery96], [Levinson94] überwacht. Für die Überwachung der Sensorsignale ist die Voraussetzung für SPC-Methoden, Normalverteilung, Unabhängigkeit und konstante Varianz der Messwerte, nicht gegeben.
Chen et al. schlagen deshalb Erweiterungen des SPC Ansatzes dahingehend vor, dass die Zeitreihen durch Benutzen eines Zweipass-Filters die Residuen der Zeitreihen bestimmt und mit Standard-Modellierungsverfahren wie EWMA (*Exponentially Weighted Moving Average*) und ARIMA (*Autoregressive Integrated Moving Average*) [Montgomery96] modelliert werden. Die Vorhersagen des Modells werden mit den tatsächlichen Messwerten verglichen. Für korrelierte Parameter wird die Verwendung von Multivariaten T^2 Kontrollkarten [Alt88], für die Überwachung unkorrelierter Parameter univariate EWMA Kontrollkarten vorgeschlagen. Für jede Recipe, Anlage und Produktkombination muss dabei ein Modell erstellt und nach Änderungen wie z.B. *Recipe*-Modifikationen, angepasst werden.

Auch Wise et al. [Wise99] verwendeten u.a. PCA und MPCA (Multiway PCA) zur Analyse der Prozessparameter. Die untersuchten Daten stammen von Trockenätzanlagen der Halbleiterproduktion. Der verwendete Prozess unterscheidet sich allerdings von den Prozessen in Kapitel 3.1 und 3.2, was bedeutet, dass die Signalverläufe nicht vergleichbar aber doch ähnlich zu den in dieser Arbeit verwendeten Zeitreihen sind. Wise et al. täuschten verschiedene Anlagenfehler wie fehlerhafte Gasflüsse und Leistungen vor und konnten mit PCA viele der Anlagenfehler als Abnormitäten in den Zeitreihen feststellen. Darüber hinaus verwendeten Wise et al. die Hotellings T^2 und Q-Statistik um die Anlagenfehler zu detektieren. Problematisch bei diesem Ansatz ist, dass bei Mehrfachverteilungen und komplexen Strukturen im Phasenraum Abnormitäten, aufgrund geringer lokaler Dichte, unter Umständen nicht erkannt werden.
Darüber hinaus ist bei einem Einsatz dieser Methoden in der Praxis zu beachten, dass zum einen ein hoher Verwaltungsaufwand in der Stammdatenpflege z.B. der Zielwerte und Spezifikationsgrenzen bei [Chen98] getrieben werden muss. Für eine hochvolumige Speicherproduktionslinie ist diese Beschränkung kein Problem, da wenige Produkte und Prozesse vorhanden sind. Für Fertigungslinien, in den ausschließlich Logikprodukte gefertigt werden, ist eine Vielzahl von bis zu mehreren hundert verschiedenen Produkten in der Linie und bis zu etlichen Dutzend verschiedenen Prozessen auf den Anlagen vorhanden. Für einen flächendeckenden Einsatz von Fehlererkennungs- und Klassifizierungsalgorithmen in üblichen Fertigungslinien mit mehr als 300 Prozessequipments stellen diese Randbedingungen dann allerdings ein ziemliches Stammdatenproblem dar.

Ein weiteres Problem der bisherigen Ansätze liegt in Mehrfachverteilungen der Daten und in der Drift des Prozesses. Die damit verbundenen Nichtstationaritäten der Sensorsignale führen zu einer regelmäßigen Nachjustierung der Modelle und einem damit verbundenen hohen Stammdaten-Pflegeaufwand. Diese Nichtstationaritäten können dabei im wesentlichen auf drei Faktoren zurückgeführt werden: Verschmutzungen durch Polymerablagerungen der Reaktorkammer zwischen den Reinigungszyklen, Unterschiede der einzelnen Wafer bedingt durch die Vorprozesse und Toleranzen, und in Drifts der Sensoren.

Zusammenfassung und Ausblick

Ziel dieser Arbeit war, die Effizienz verschiedener Netzwerkmodelle und –architekturen zur Analyse und Klassifikation hochdimensionaler, multisensorischer Datensätze prozessierter Si-Wafer zu untersuchen. Eine Aufgabenstellung bestand darin, Abnormitäten im hochdimensionalen Raum der multisensorischen nichtlinearen Zeitreihen zu detektieren und ggf. eine Klassifizierung zu ermöglichen. Die Datenanalyse wird aufgrund partiell fehlenden Vorwissens sowohl überwacht als auch mit unüberwachten Methoden durchgeführt. Um den „Fluch der Dimensionalität“ zu vermeiden sowie systeminhärente Information zu detektieren, wird der hochdimensionale Datenraum reduziert bzw. signifikante Kennzahlen extrahiert, um anschließend eine einfache Klassifikation dieser Daten zu ermöglichen.
Der zweite Teil der Arbeit bestand in der Analyse und Klassifizierung hochdimensionaler Datenräume ohne Dimensionsreduzierung. Anhand mehrdimensionaler PCM-Messwerte und der zugehörigen Ausbeuteinformation werden Analysen dieses Datenmaterials durchgeführt und verschiedene Ansätze zur Klassifikation untersucht.

Die Auswertungen mit Methoden der nichtlinearen Zeitreihenanalyse zeigen, dass Autokorrelationen in den Datenreihen enthalten sind und eine ausreichend hohe Samplingrate gewählt wurde. Kreuzkorrelationen zwischen den einzelnen Signalverläufen sind vorhanden und können teilweise zur Reduktion des Datenraumes verwendet werden. Aufgrund der Begrenzung der Schnittstellenkapazität werden möglichst unabhängige Parameter erfasst, um so die zugrundeliegende Dynamik optimal modellieren zu können.
Zur Abschätzung der Komplexität bzw. der Dimensionalität für die Modellierung wurde die Einbettungsverzögerung τ_E unter Verwendung der *Average Mutual Information* und Dimensionalität des rekonstruierten Attraktors mit der *false-nearest-neighbor*-Methode bestimmt. Diese Abschätzungen können als Grundlage zum Netzwerkdesign verwendet werden [Kulkarni97]. In den untersuchten Zeitreihen liegt die Einbettungsverzögerung bei 13 Sekunden und die Einbettungsdimension im Bereich zwischen 2 und 6 Dimensionen.

Für den Fall großer Datenmengen ohne implizite Klasseneinteilung zeigte die Dimensionsreduzierung der nichtlinearen Zeitreihen durch PCA, Analysen mit ICA und Klassifikation durch Kohonennetze bereits im zweidimensionalen Raum deutliche Cluster und einige wenige Ausreißer, d.h. Gebiete mit niedriger Population im Phasenraum. Diese Neuronen wurden zur Detektion von Ausreißern bzw. abnormalen Gebieten verwendet.
Eine gut- bzw. schlecht-Klassifikation, kann aufgrund dieser Bewertung allerdings nicht durchgeführt werden. Der Rückschluss von abnormal zu schlechtem Prozessergebnis hängt

stark von den Prozesstoleranzen ab und kann nur in der Analyse bzw. Korrelation der Zeitreihen mit den Prozessergebnissen mit signifikanter Wahrscheinlichkeit angegeben werden.

Für den Fall einer Normalprozessierung besteht eine große Wahrscheinlichkeit für eine gute Prozessierung, im Sinne der Erreichung des Prozesszieles. Konkret für das Beispiel von 90% Ausbeute muss die Wahrscheinlichkeit für eine gute Prozessierung pro Prozessschritt bei $0{,}9^{1/400} = 0{,}997\%$ liegen. Anders formuliert heißt das, dass für die Trockenätzungen, die in dieser Arbeit untersucht wurden, im statistischen Mittel eine Fehlerrate kleiner als 0,3% vorliegen muss, was bei der vorliegenden Datenmenge von 1695 Wafer ca. 5 Wafer entspricht.

Phasenraumbereiche niedriger Dichte wurden damit als kritische Bereiche, sei es als qualitätsrelevant oder auch produktivitätsrelevant im Sinne ungeplanter Störungen der Anlage betrachtet. Diese Annahme wurde sowohl von Ergebnissen dieser Arbeit als auch von anderen Arbeiten gestützt [Wise99], [Chen98], u.a. Eine Prozesscharakterisierung durch die Prozessparameter in Phasenräumen niedriger Dichte weisen damit auf bevorstehende Probleme von Anlagenteilen wie z.B. Lecks in den Vakuumkammern hin und sind trotz der nicht eindeutig zuordenbaren gut/schlecht-Klassifizierung von Interesse.

Die multivariate Betrachtung der Hauptkomponenten der Prozessparameter zeigt komplexe Strukturen im mehrdimensionalen Raum, was auf ein hochdimensionales Klassifikationsproblem hinweist. Aus diesem Grund wurden zwei Methoden zur Ausreißerbestimmung verwendet, die im mehrdimensionalen Gültigkeit besitzen, eine Analysegeschwindigkeit zulassen, die eine *in-situ*-Analyse ermöglichen und Anlagendrifts berücksichtigen können.
Die verwendeten Methoden basieren auf der Bestimmung der lokalen Neuronendichte durch Rasterung der Kohonenkarte in n-dimensionale Quader. Dieses Vorgehen erinnert sehr stark an RCE-Klassifikationsalgorithmen (*Reduced Coulomb Energy*) [Reilly82], [Moreno95]. Qualitativ kann dort zumindest die Aussage getroffen werden, dass die Dichteverteilung in den Phasenraumsegmenten indirekt proportional zur Wahrscheinlichkeit ist, dass diese Neuronen abnormale Wafer detektieren.

Die anschließenden Auswertungen mit Backpropagation-Netzen mit einer verdeckten Schicht und einem Ausgabeneuron zeigen, dass diese Netze gut geeignet sind, die verschiedenen Zeitreihenverläufe aufgrund der *a-priori*-Klasseneinteilung zu separieren. Ein großer Vorteil dabei ist, dass zwar mit einer digitalen Klasseneinteilung trainiert wird, aber aufgrund der unterschiedlichen Ausprägung des Problems das Netzwerk in der Lage ist, die Klassenteilung in einen kontinuierlichen Bereich zu projizieren. Die gut/schlecht Aussage relativiert sich damit zu einer Art Fuzzy Logik für die Wahrscheinlichkeit der Klassenzugehörigkeit und bietet dem Anwender eine gute Entscheidungshilfe. Problematisch ist allerdings, dass aufgrund der Separation des Phasenraumes durch Hyperebenen es unter gewissen Umständen zu Fehlklassifikationen vorkommen kann, wie im Beispiel mit den Sinuskurven im Abschnitt 3.2.6 gezeigt wird.

Zusätzlich zur ursprünglichen Aufgabenstellung eröffnete sich ein Verfahren zur Analyse der Trainingsdaten zur Untersuchung auf deren Relevanz für das zu untersuchende Problem. Diese Option ist insofern interessant, da die Prozessanlagen mehrere hundert Parameter potentiell erfassen könnten, aufgrund der Einschränkungen in der Datenübertragung allerdings eine Selektion vorgenommen werden muss. Darüber hinaus sind viele Prozessvariablen noch nicht richtig verstanden. Die Information über die Relevanz bestimmter Parameter zur Fehlerdetektion ist eine Voraussetzung für den Erfolg nachgeschalteter Fehlererkennungs- bzw. Klassifikationsalgorithmen. Durch die Analyse der Synapsen des neuronalen Netzes können sowohl die Trainingsdaten als auch bestimmte Kurvenabschnitte beurteilt werden. Dies kann dazu eingesetzt werden, um ausgehend von der Information, dass eine Prozessierung normal bzw. abnormal ist, die verantwortlichen Parameter und *Steps* aus den Prozessierungsdaten zu ermitteln.

Zur anschließenden Mustererkennung und Klassifizierung hochdimensionaler Datensätze wurden verschiedene neuronale Netzwerke untersucht. Das verwendete Datenmaterial stammt von Testmessungen unmittelbar nach der Prozessierung der Scheiben im *Frontend*. Die Bewertung dieser Datensätze erfolgte anhand der Ausbeute der entsprechenden Wafer. Mit diesen Datensätzen war es möglich, verschiedene Klassifikationsansätze zu untersuchen und weitere Ergebnisse über funktionelle Zusammenhänge von Datenstrukturen prozessierter Si-Wafer zu finden.

Die Projektion des 22-dimensionalen Datenraumes auf zwei Dimensionen lassen vorhandene Cluster vermuten. Durch die induzierte Delaunay-Triangulation wird gezeigt, dass es sich tatsächlich um Cluster handelt. Untersuchte Cluster zeigen eine Homogenität bzgl. der Ausbeute und eine auf das verwendete euklidische Distanzmaß bezogene Ähnlichkeit. Für die untersuchten PCM-Daten der Si-Wafer ergeben sich damit Bereiche im Phasenraum mit ähnlichen Eigenschaften und zugehörigen Referenzvektoren. Diese können für den Fall niedriger Ausbeuten auch als typische Ausfallmuster definiert werden. GNG-Netze (*Growing Neural Gas*) sind dabei in der Lage, entsprechende Muster bzw. Cluster zu finden. Ein Vergleich mit Kohonenkarten zeigt, dass der quadratische Fehler für GNG, bei gleicher Anzahl verwendeter Neuronen, geringer ist.

Für überwacht wachsende neuronale Gase [Fritzke94] ergaben sich keine wesentlichen Änderungen bzgl. des Lernfehlers. Dies kann daran liegen, dass die gesuchte Abbildung des Eingaberaumes auf die Ausbeute zu komplex ist, um sie mit dieser, relativ einfachen, Erweiterung für SGNG zu verbessern. Da die Ergebnisse bei GNG bereits eine Homogenität bzgl. der Ausbeute in den gefundenen Clustern aufweisen, ist dieses Ergebnis jedoch als weiteres Indiz dafür zu werten, dass die Separation des Eingaberaumes und Erstellung eines Codebuches bereits durch GNG zufriedenstellend gelöst wurde.

Lokal konstante Abbildungen zerlegen den Phasenraum in n zusammenhängende Unterräume in denen der zugehörige Ausgabewert durch eine skalare Größe, üblicherweise den arithmetischen Mittelwert, genähert wird. Entscheidend für eine gute Abbildung ist die Wahl

der Unterräume. Die Klasseneinteilung innerhalb der Unterräume sollte dabei möglichst homogen sein. Diese Eigenschaft erfüllen die Voronoigebiete der wachsenden Netze und werden deshalb, neben der Bestimmung durch Kohonenkarten und einfacher zufälliger Auswahl von Testmustern, zur Bestimmung des Codebuches herangezogen und verglichen.
Die besten Ergebnisse wurden dabei für die wachsenden neuronalen Gase erreicht. Das SGNG lernt dabei schneller, zeigt aber auch eher Zeichen von *Overfitting*. Die besten Ergebnisse wurden für etwa 200 – 400 Neuronen erreicht. Im Mittel liegen dann fünf Trainingsmuster im Voronoigebiet jedes Neurons. Die Qualität der mit dem Kohonen-Verfahren erstellten Codebücher ist dabei besser als angenommen. Mit einem mittleren Klassifizierungsfehler von 3,4% erreichten damit die überwachten neuronalen Gase mit lokal konstanten Abbildungen bei einer Größe von 200 – 400 Neuronen die besten Ergebnisse. Dieser Wert liegt deutlich unter der Differenz nächster Nachbarn von 3,81%.

Bei den überwachten wachsenden neuronalen Gasen werden neue Neuronen mit dem Ziel eingefügt, den lokalen Klassifizierungsfehler zu minimieren. Das bedeutet, dass neue Referenzvektoren dort eingefügt werden, wo die Streuung der Klassenzugehörigkeit groß ist. Dem Klassenlabel der Neuronen entspricht dabei ungefähr die mittlere Ausbeute der zugehörigen Voronoigebiete. Die wachsenden, lokal konstante Abbildungen (GLCM) nutzen diese bereits während des Trainingsvorganges vorhandene Klasseninformation. Während des Wachstums können zu beliebigen Zeitpunkten Testmuster direkt klassifiziert werden, indem jedem Testmuster einfach der Klassenlabel des jeweiligen Siegerneurons zugeordnet wird.

Der entscheidende Vorteil dieser Erweiterung von überwachten wachsenden neuronalen Gasen ist die mögliche Klassifizierung und Überwachung von nichtstationären Verteilungen. Für die Daten der Prozessierten Si-Wafer bedeutet das, dass jeder neu gemessene Wafer zunächst klassifiziert, also eine Aussage über das zu erwartende Ergebnis getroffen, wird. Die anschließende Bewertung durch *Inline*-Messungen oder Funktionstestmessungen, können die Gewichte und Label der Siegerneuronen nach den bekannten Schemata aktualisiert und der nichtstationäre Charakter der zugrundeliegenden Systemdynamik berücksichtigt werden.

Im Vergleich von RBF-Netzen mit LCM-Verfahren weisen die RBF-Netze für alle Netzwerkgrößen eine um 1,5% schlechtere Klassifikationsleistung auf. Zum Nachweis, dass diese Form von RBF-Netzen in der Lage sind, typische Klassifikationsprobleme zufriedenstellend zu lösen, wurde das *vowel-recognition*-Problem aus der Serie der *CMU-Benchmarks* untersucht. Die Zentren der verdeckten Schicht wurden durch wachsende neuronale Gase ermittelt. Für diese Aufgabenstellung lag der Anteil korrekt klassifizierter Muster mit etwa 65% deutlich über den in der Literatur berichteten Ergebnissen, womit gezeigt ist, dass die Klassifizierung von PCM-Daten eine Herausforderung an Klassifikatoren darstellt.

Insgesamt scheint bei etwa 3,4% Klassifikationsfehler der PCM-Daten eine Grenze erreicht zu sein, die durch die Struktur der Daten vorgegeben ist und nicht unterschritten werden kann. Für einen tatsächlichen Einsatz der vorgestellten Methoden ist vor allem wichtig, dass die größeren Klassifikationsfehler stets im Bereich geringer Ausbeuten gemacht wurden. So

würde häufiger ein Wafer mit geringer Ausbeute zu hoch eingeschätzt (falsche Akzeptanz) als dass ein Wafer mit hoher Ausbeute fälschlicherweise vorzeitig aussortiert würde (falsche Zurückweisung).

Zusammenfassend kann festgestellt werden, dass in den multisensorischen Zeitreihen und mehrdimensionalen Testdaten sehr viel Information über den Prozess- bzw. Anlagenzustand enthalten ist. Unüberwachte Analysemethoden extrahieren systeminhärente und zur Klassifikation relevante Information über den Prozessverlauf. Überwachte neuronale Netze extrahieren Kennzahlen für spezielle Fehlerbilder und erzeugen eine Aussage über die Relevanz von Prozessparametern bzw. deren Rezeptschritte für bestimmte Fehlerbilder. Zur Klassifikation dieser Daten müssen multivariate Methoden verwendet werden. Die Problematik bisher in der Literatur beschriebenen Methoden liegt darin, dass diese die Nichtstationarität der Daten nicht ausreichend berücksichtigen und aufgrund der großen Stammdatenpflege nicht für Logikfabriken einsetzbar sind. Durch den Einsatz selbstadaptiver Verfahren zur Kennzahlenextraktion und Klassifikation können diese Probleme umgangen und damit als Ergänzung zum bisherigen SPC-Ansatz eine 100% Prozesskontrolle ermöglicht werden. Die Nutzung des Potentials, durch Auswertung der multisensorischen Zeitreihen, von geschätzten 10% Produktivitätssteigerung und 3% Ausbeutesteigerung, ist damit in greifbare Nähe gerückt.

Literatur

[Abarbanel93] Abarbanel HDI, Brwon R, Sidorowich JJ, Tsimring LS. The Analysis of observed chaotic data in physical systems, Rev. Mod. Phys. 65, 1331, 1993

[Alt88] Alt FB, Smith ND. Multivariate Process Control. Handbook of Statistics Vol.7, Elsevier Science Publ. BV, 333-351,1988

[Balkin00] Balkin SD, Ord JK. Automatic neural network modeling for univariate time series. Int.J.Fore.16, 509-515, 2000

[Barrios98] Barrios lJ, Lisstette L. Autoassociative Neural Networks for Fault Diagnosis in Semiconductor Manufacturing. Lecture Notes in Comp. Sci., 582-592, 1998

[Bartlett98] Bartlett MS, Lades HM, Seynowski TJ. Independent component representations for face recognition. Proc. SPIE Symp. Electr. Imaging, San Jose CA, Januar, 1998

[Berthold99] Berthold M, Hand DJ. Intelligent Data Analysis. Springer, 1999

[Bhatikar99] Bhatikar S, Mahajan RL. Neural network based diagnosis of CVD barrel reactor. Advances in Elec.Pack., Proc. ASME InterPACK, Vol.26, 621-640, 1999

[Bishop95] Bishop CM. Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1995

[Bishop98] Bishop CM. Neural Networks and Machine Learning, Springer, 1998

[Bothe98] Bothe HH. Neuro-Fuzzy Methoden, Einführung in Theorie und Anwendungen. 1998

[Bronstein91] Bronstein IN, Semendjajew KA. Taschenbuch der Mathematik. Teubner, 1991

[Bruske94] Bruske J, Sommer G. Dynamic Cell Structures. Proceedings NIPS 7, 497-504, 1994

[Bruske95] Bruske J, Sommer G. Dynamic Cell Structure Learns Perfectly Topology Preserving Map. Neural Computation 7, 845-865, 1995

[Bushmann97] Bushmann S, Edgar TF, Trachtenberg I, Modeling of Plasma Etch Systems using Ordinary Least Squares, Recurrent Neural Network, and Projection to Latent Structure Models. J. Elect. Soc, Vol.144, No.4, 1997

[Card00] Card JP. A Study in Dynamic Neural Control of Semiconductor Fabrication Processes. IEEE Trans. Semi. Manu., Vol.13, No.3, 2000

[Cardoso97] Cardoso JF. Infomax and maximum likelihood for blind source separation. IEEE Sig.Proc.Let. 4, 109-111, 1997

[Cheeseman88] Cheeseman P, Freeman D, Kelly J, Self M, Stutz J, Taylor W. AutoClass: a Bayesian classification system. Proc Fifth Inter Conf Machine Learning, 1988

[Cheeseman89] Cheeseman P, Stutz J, Self M, Taylor W, Goebel J, Volk K, Walker H. Automatic classification of spectrum from the infrared astronomicla satellite. NASA reference publication #1217; National technical Information Service, Springfield, VA, 1989

[Cheeseman96] Cheeseman P, Stutz J. Bayesian classificatioin (AutoClass): theory and results. AAAI Press/MIT Press, 153-180, 1996

[Chen98] Chen A, Guo RS, Yang A, Tseng CL. An Integrated Approach to Semiconductor Equipment Monitoring. J. Chinese Soc. Mech. Eng., Vol.19, No.6, 581-591, 1998

[Chen00] Chen FL, Liu SF. A Neural-Network Approach To Recognize Defect Spatial Pattern In Semiconductor Fabrication. IEEE Trans. Semi. Manu. Vol.13, No.3, 2000

[Dietrich95] Dietrich E, Schulze A. Statistische Verfahren zur Maschinen- und Prozessqualifikation. Hanser Vlg. 1995

[Domany94] Domany E, Hemmen van JL, Schulten K. Models of Neural Networks II. Springer, New York, 1994

[Everitt81] Everitt BS, Hand DJ. Finite mixture distributions. London Chapman&Hall, 1981

[Fahrmeir96] Fahrmeir L, Hamerle A, Tutz G. Multivariate statistische Verfahren. Walter de Gruyter, 1996

[Fischer99] Fischer TM. Datenanalyse und Fehlerklassifikation mit neuronalen Netzen. Diplomarbeit FH Regensburg, 1999

[Fraser86] Fraser AM, Swinney HL, Independent coordinates for strange attractors from mutual information. Phys.Rev. A 33, 1134, 1986

[Fritzke93] Fritzke B. Growing Cell Structures-A self organizing Network for Unsupervised and Supervised Learning, ICSI, 1993

[Fritzke94] Fritzke B. Fast learning with incremental RBF Networks. Neural Processing Letters 1, 2-5, 1994

[Fritzke95] Fritzke B. A Growing Neural Gas Network Learns Topologies. Advances in Neural Inf. Proc. Syst., MIT Press, Cambridge MA, 1995

[Fritzke98] Fritzke B. Vektorbasierte neuronale Netze. Habilitation. Shaker Vlg. 1998

[Fritzke99] Fritzke B. A self-organizing network that can follow non-stationary distributions. Proc. ICANN97, 613-618, 1999

[Ganslmeier99] Ganslmeier B. Neuronale Netzwerkmodelle zur Analyse hoch-dimensionaler, multisensorischer Datensätze bei der Wafer-Prozessierung. Diplomarbeit Univ. Regensburg, 1999

[Ganslmeier00] Ganslmeier B, Schels A, Lang EW. PCA and ICA Analysis of Process Control Data obtained during SI-Wafer Manufacturing. Proc. SPC 2000, Marbella Spain, 2000

[Ghaz97] Ghazanfarian AA, Pease RFW, Chen X, CmCord MA. Neural network model for global alignment incorporating wafer and stage distortion. J.Vac.Sci.Tech. B15(6), 2146-2150, 1997

[Girolami97] Girolami M, Fyfe C. Extraction of Independent Signal Sources using a Deflationary Exploratory Projection Pursuit Network with Lateral Inhibition. IEE Proc.Vis. Image Sig.Proc. J. Vol14, 299-306, 1997

[Girolami99] Girolami M. Self-Organising Neural Networks. Springer London, 1999

[Grossberg76] Grossberg S. Adaptive pattern classification and universal recording: I parallel development and coding of neural feature detector. Cybernet, 23, 121-134, 1976

[Grossberg76a] Grossberg S. Adaptive pattern classification and universal recording: II feedback, expectatioin, olfaction, illusions. Cybernet 23, 187-202, 1976

[Haken83] H. Haken. Synergetik. Springer, Berlin, 1983

[Haykin99] Haykin S. Neural Networks – A Comprehensive Foundation. MacMillan College Pub. 1999

[Hertz91] Hertz J, Krogh A, Palmer RG, Introduction to the theory of Neural Computation, 1991

[Hotelling33] Hotelling H. Analysis of a compex of statistical variables into principal components. J Educ. Psychol. 24, 417-441, 498-520, 1933

[Huber85] Huber PJ. Projection Pursuit, Annals of Statistics 13, 2, 435-475, 1985

[Jäger00] Jäger O. Analyse und Klassifikation von Fluoreszensspektren von Blutzellen mit neuronalen Netzen. Diplomarbeit, Univ. Regensburg, 2000

[Jutten91] Jutten C, Herault J. Blind separation of sources, part I: An adaptive algorithm based on neuromimetric architcture. Sig. Proc. 24, 1-10, 1991

[Karhunen96] Karhunen J. Neural Approaches to Independent Component Analysis and Source Separation. Helsinki Univ. Tech., 1996

[Kendall69] Kendall MG, Stuart A. The advanced theory of statistics, Vol.1, Charles Griffin, London, 1969

[Kennel92] Kennel MB, Brown R, Abarbanel HD. Determining embedding dimension for phase-space reconstruction using geometrical construction. Phys.Rev. A45, 3403, 1992

[Kohonen82] Kohonen T. Self-Organized formation of topologically corret feature maps. Biol. Cyb. 43, 59-69, 1982

[Kohonen90] Kohonen T. The self-organizing map. Proc. IEEE78, 1464-1480,1990

[Kulkarni97] Kulkarni DR, Parikh JC. Dynamic Predictions From Time Series Data - An Artificial Neural Network Approach. Int. Journal of Modern Physics C, Vol.8, No.6, pp.1345-1360, 1997

[Leim00] Leim D. Neuronale Netzwerkmodelle zur Mustererkennung und Klassifizierung hochdimensionaler Testdatenfelder von Prozessierten Si-Wafern. Diplomarbeit, Univ. Regensburg, 2000

[Leger98] Leger RP, Garland WJ, Poehlman WFS. Fault detection and diagnosis using statistical control charts and artificial neural networks. Art.Intel.Eng. 12, 35-47, 1998

[Levinson94] Levinson, W.A. Statistical process control in microelectronics manufacturing. Semiconductor International, Nov. 1994

[Lim99] Limanond S, Si J, Tseng YL. Production data based optimal etch time control design for a reactive ion etching process. IEEE Trans. Semi. Manu., Vol.12, No.1, 1999

[Luenb71] Luenberger DG. An Introduction to Observers. IEEE Trans.Aut.Control, Vol.16, No.6, 596-602, 1971

[Malthouse98] Malthouse ED. Limitations of Nonlinear PCA as Performed with Generic Neural Networks. IEEE Trans. Neural Networks, Vol.9, No.1, 1998

[Martinetz91] Martinetz TM, Ritter HJ, Schulten KJ. A neural-gas network learns topologgies. Art. Neural Networks, 397-402, 1991

[Meier00] Meier G. Untersuchung nichtlinearer Zeitreihenanalysemethoden. Diplomarbeit Univ. Regensburg, 2000

[Moody89] Moody J, Darken C. Learning with localized receptive fields. Proc. Conn. Models Summer School, 133-143, San Mateo, 1989

[Moreno95] Moreno JM, Vazques FX, Castillo F. A deterministic Mehod for establishing the initial conditions in the RCE algorithm. Europ.Sympos. on Art. Neural Networks (ESANN), 1995

[Montgomery96] Montgomery DC, Introduction to Statistical Qualtiy Control, John Wiley and Sons, 1996

[Oja82] Oja E. A simplified neuron model as a principal component analyzer. J. of Math. Biol. 15, 267-273, 1982

[Oja95] Oja E. The Nonlinear PCA Learning Rule and Signal Separation – Mathematical Analysis. Helsinki Univ. Tech. Report A26, 1995

[Patterson96] Patterson DW. Künstliche Neuronale Netze. Prentice Hall, 1996

[Pearson01] Pearson K. On lines and planes of closest fit to systems of points in space. Phi.1 Mag. 2, 559-572, 1901

[Press98] Press W, Flannery B, Teukolsky S, Vetterling W. Numerical Recipes in C. Cambrigde Univ. Press, 1998

[Reily82] Reilly DL, Cooper LN Elbaum C. A neural model for category learning. Biological Cybernetics 45, 35-41, 1982

[Rietmann01] Rietmann EA, Whitlock SA, Beachy M, Roy A, Willingham TL, A system model for feedback control and analysis of Yield. IEEE Trans Semi Manu., Vol.14, No.1, 2001

[Ritter91] Ritter H, Martinetz T, Schulten K. Neuronale Netze. Addison Wesley 1991

[Robinson89] Robinson AJ. Dynamic Error Propagation Networks. Ph.D. Thesis, Cambridge Univ. Eng. Dep., 1989

[Rohatsch99] Rohatsch T. Multidimensionale Klassifikation durch neuronale Netze. Diplomarbeit FH Regensburg, 1999

[Rudolph99] Rudolph A. Statistische Verfahren der Klassifikation. Shaker, 1999

[Salam97] Salam FM et al. Modeling of a Plasma Processing Machine for Semiconductor Wafer Etching Using Energy-Functions-Based Neural Networks. IEEE Trans. Control Sys., Vol.5, No.6, 598-613, 1997

[Sanger89] Sanger TD. Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Networks 12, 459-473, 1989

[Schmalzb01] Schmalzbauer M. Vorhersage von Prozessparametern zur Online-Fehlererkennung. Diplomarbeit FH Regensburg, 2001

[Shannon48] Shannon C. A mathematical theory of communication. Bell Sys. Tech. J. 27, 379-423, 1948

[Shewhart31] Shewhart WA. Economic Control of Quality of Manufactured Product. Van Nostrand, New York, 1931

[Takens97] Takens F. Detecting strange attractors in turbulence. Lecture notes in math., 366, Springer, Berlin, 1997

[Thomson92] Thomson RF. Das Gehirn. Akademischer Verlag, Heidelberg, 1992

[Wang95] Wang XA, Mahajan RL. CVD epitaxial deposition in a vertical barrel reactor, process modelling and optimization using neural network models. J. Electrochem. Soc. Vol.142, No.2, 1323, 1995

[Wang96] Wang XA, Mahajan RL. Artificial neural network model-based run-to-run process controller. IEEE Trans. Comp. Pack. Manu. Tech.Part C, Vol.19, No.1, 19-26, 1996

[Wang99] Wang XZ. Data Mining and knowledge discovery for process monitoring and control. Springer, 1999

[Weigend94] Weigend AS, Gershenfeld NS. Time Series Prediction, Addison-Wesley, Mass, 1994

[West99] West DA, Mangiameli PM, Chen SK. Control of compelex manufacturing processes: a comparison of SPC methods with a radial basis function neural network. Omega, Int.J.Mgmt.Sci.27, 349-362, 1999

[Wise99] Wise BM, Gallagher NB, Butler SW, White DD, Barna GG. A Comparison of Principal Component Analysis, Multiway Principal Component Analysis, Trilinear Decomposition and Parallel Factor Analysis for Fault Detection in a Semiconductor Etch Process. J. Chemometrics, Vol. 13, p.379-396, 1999

[Whitney36] Whitney H. Differentiable manifolds. Ann. Math. 37, 645, 1936

[Yang97] Yang HH, Amari S. Adaptive On-Line Learing Algorithms for Blind Separation – Maximum Entropy and Minimum Mutual Information. Neural Computation, 1997

[Zell97] Zell A. Simulation neuronaler Netze. 1997

[Zhang01] Zhang GP, Patuwo BE, Hu MY. A simulation study of artificial neural networks for nonlinear time-series forecasting. Comp.Oper.Res.28, 381-396, 2001

[Zorrias98] Zorraisstatine F, Tannock JDT. A review of neural networks for statistical process control. J. Intel. Manu., 9, 209-224, 1998