J.W.Goethe-Universität
Frankfurt am Main

Institut für Informatik
Prof. Dr. Rüdiger Brause

Nonlinear Feature Selection using the General Mutual Information

Dipl.-Inform. Frank Heister

Satz

Reproduktionsfähige Vorlage vom Autor.
Gesetzt in LaTeX.

vom Fachbereich Informatik und Mathematik der

Johann Wolfgang Goethe – Universität als Dissertation angenommen.

Dekan: Prof. Dr. K. Johannson

1. Gutachter: Prof. Dr. R. Brause

2. Gutachter: Prof. Dr.-Ing. R. Mester

Vorsitz: Prof. Dr. K. Waldschmidt

Protokoll: Prof. Dr. I. Timm

Datum der Disputation: 11. April 2008

Nonlinear Feature Selection using the General Mutual Information

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe – Universität
in Frankfurt am Main

von

Dipl.-Inform. Frank Heister

aus

Wunsiedel im Fichtelgebirge

Frankfurt am Main, 2008

Meinen lieben Großeltern,
Karl und Liselotte Heister,
in tiefer Dankbarkeit und Liebe.

Danksagung

Die ursprüngliche Idee zu der vorliegenden Arbeit entstand im Rahmen meiner Tätigkeit in der Forschungsabteilung der DaimlerChrysler AG. Da das Leben aber immer die eine oder andere unvorhersehbare Prüfung für uns bereit hält, mußte ich die Fertigstellung der Arbeit leider auf einen späteren Zeitpunkt verschieben. Im Laufe dieser Zeit wurde ich bei der Anfertigung der Arbeit von einer Reihe lieber Menschen begleitet und unterstüzt, denen ich an dieser Stelle herzlich danken möchte.

Mein besonderer Dank gilt Herrn Prof. Dr. Brause, der mich zur Fortführung meiner Arbeit ermutigte und mich stets mit wertvollen Anregungen und mit seiner konstruktiven Sichtweise unterstüzte.

Bei meinen beiden Diplomanden Sven Wahler und Alberto Ginestroni möchte ich mich an dieser Stelle ebenfalls herzlich bedanken. Die muttersprachlichen Fähigkeiten von Alberto Ginestroni erleichterten deutlich die Kommunikation mit Professor Alessandro De Carli von der Universität La Sapienza in Rom. Mein Dank gilt auch meinen ehemaligen Kollegen Michael Fröhlich und Gregor Schock, auf deren Unterstützung ich immer zählen konnte.

Ich bedanke mich ganz herzlich bei allen Menschen die am *Gelingen* dieser Arbeit mitwirkten. Besondere Unterstützung während der Arbeit fand ich bei meiner Familie und bei meinen Freunden. Sie ertrugen mit Fassung und Liebe, manchmal aber auch mit einem beherzten Schmunzeln, meine arbeitsbedingten Hoch- und Tiefpunkte und gaben mir dadurch die notwendige Kraft und Ausdauer.

And now, the end is near;
And so I face the final curtain.
My friend, I'll say it clear;
I'll state my case, of which I'm certain.

I've lived a life that's full;
I've travelled each and ev'ry highway.
And more, much more than this;
I did it my way.

Regrets, I've had a few;
But then again, too few to mention.
I did what I had to do;
And saw it through without exemption.

I planned each charted course;
Each careful step along the byway.
And more, much more than this;
I did it my way.

Yes, there were times, I'm sure you knew;
When I bit off more than I could chew.
But through it all, when there was doubt;
I ate it up and spit it out.
I faced it all and I stood tall;
And did it my way.

I've loved, I've laughed and cried;
I've had my fill; my share of losing.
And now, as tears subside;
I find it all so amusing.

To think I did all that;
And may I say - not in a shy way;
"Oh no, oh no not me,
I did it my way".

For what is a man, what has he got?
If not himself, then he has naught.
To say the things he truly feels;
And not the words of one who kneels.
The record shows, I took the blows -
And did it my way!

(Frank Sinatra)

Contents

1 Zusammenfassung

Der Begriff der Transinformation wurde erstmals von Claude Elwood Shannon im Kontext der Informationstheorie, einer einheitlichen mathematischen Beschreibung technischer Kommunikationssysteme, geprägt. Die Informationstheorie bildet heute die Grundlage vieler Anwendungen der modernen Kommunikations- und Nachrichtentechnik und ist aus diesen Bereichen nicht mehr wegzudenken.

Die vorliegenden Arbeit befasst sich vor diesem Hintergrund mit der Entwicklung einer in der Praxis anwendbaren Methodik zur nichtlinearen Merkmalselektion quantitativer, multivarianter Daten auf der Basis des bereits erwähnten informationstheoretischen Ansatzes der Transinformation. Der Erfolg beim Übergang von realen Messdaten zu einer geeigneten Modellbeschreibung wird maßgeblich von der Qualität der verwendeten Datenmengen bestimmt. Eine qualitativ hochwertige Datenmenge besteht im Idealfall ausschließlich aus den für eine erfolgreiche Modellformulierung relevanten Daten. In diesem Kontext stellt sich daher sofort die Frage nach der Existenz eines geeigneten Maßes, um den Grad des, im Allgemeinen nichtlinearen, funktionalen Zusammenhangs zwischen Ein- und Ausgaben quantitativ korrekt erfassen zu können. Mit Hilfe einer solchen Größe können die relevanten Merkmale gezielt ausgewählt und somit von den redundanten Merkmalen getrennt werden. Im Verlaufe dieser Arbeit wird deutlich werden, dass die eingangs erwähnte Transinformation ein hierfür geeignetes Maß darstellt und im praktischen Einsatz bestens bestehen kann.

Ohne den Anspruch auf Vollständigkeit, existieren im wesentlichen zwei Situationen, die eine Merkmalselektion motivieren. Einerseits wird eine Merkmalselektion dann notwendig, wenn die praktische Berechenbarkeit von Computermodellen aufgrund der großen Anzahl von vorhandenen Merkmalen nicht mehr gewährleistet ist. Andererseits macht das Vorhandensein einer eingeschränkten Anzahl von Daten, mit einer Vielzahl von

Merkmalen, ebenfalls eine Merkmalselektion notwendig. Im Laufe dieser Arbeit werden Beispiele und Lösungen für beide Situationen vorgestellt. Letztere Situation ist sehr eng mit dem sogenannten "Fluch der Dimensionen" verknüpft. Dahinter steckt letztendlich die Aussage, dass bei einer eingeschränkten Anzahl verfügbarer Daten eine Reduktion deren Dimension erfolgen muß. Der Grund hierfür ist die Tatsache, dass bei einer festen Anzahl von Datenpunkten die Überdeckungsdichte des Datenraumes exponentiell mit dessen Dimension sinkt. Im Rahmen der Abbildung zwischen Ein- und Ausgaberaum erreicht man dieses Ziel idealerweise durch Selektion der, im betrachteten Kontext, wichtigen Merkmale.

Die ursprüngliche Motivation zur Erstellung der vorliegenden Arbeit hat ihren durchaus praktischen Hintergrund in der Automobiltechnik. Sie entstand im Rahmen eines komplexen Forschungsprojektes zur Ermittlung von nichtlinearen, dynamischen Zusammenhängen zwischen einer Vielzahl von messtechnisch ermittelten Sensorsignalen. Das Ziel dieser Aktivitäten war, durch die Identifikation von nichtlinearen, dynamischen Zusammenhängen zwischen den im Automobil verbauten Sensoren, sogenannte virtuelle Sensoren abzuleiten. Der Unterschied zwischen virtuellen- und realen Sensoren liegt in der Tatsache, dass virtuelle Sensoren selektiv auf bestimmte messtechnische Problemstellung hin adaptiert werden können. Die konkrete Aufgabenstellung bestand nun darin, die Bestimmung einer zentralen Motorgröße so effizient zu gestalten, dass diese ohne zusätzliche Hardware unter harten Echtzeitvorgaben berechenbar ist. Auf den zusätzlichen Einsatz von Hardware verzichten zu können und mit der bereits vorhandenen Rechenleistung auszukommen, stellt aufgrund des resultierenden, enormen Kostenaufwandes, insbesondere in der Automobiltechnik, eine unglaublich starke Motivation dar. In diesem Zusammenhang trat immer wieder die große Problematik zutage, eine praktisch berechenbare Methode zu finden, die sowohl lineare- als auch nichtlineare Zusammenhänge zuverlässig quantitativ erfassen kann.

Wie eingangs bereits erwähnt, ist das Ziel dieser Arbeit die Entwicklung einer flexibel anwendbaren Methodik zur nichtlinearen Merkmalselektion. Ein wesentlicher Punkt hierbei ist die Gewährleistung der praktischen Berechenbarkeit der entwickelten Methode, im Besonderen für hochdimensionale Datenräume. Die Notwendigkeit zur Entwicklung der genannten Methode ist ebenfalls durch die Tatsache motiviert, dass in realen technische Anwendungen hochdimensionale Datenräume eher die Regel denn die Ausnahme sind.

Um nun dieses Ziel zu erreichen, werden bereits existierende Methoden miteinander kombiniert und in geeigneter Weise aufeinander abgestimmt. Das verwendete Hautpkriterium zur Merkmalselektion basiert auf dem informationstheoretischen Ansatz der Transinformation. Die Eigenschaft der Transinformation, sowohl hinsichtlich linearer- als auch hinsichtlich nichtlinearer statistischer Zusammenhänge, hohe Sensitivität und Spezifität zu besitzen, macht diese zu einem hervorragenden Kriterium für die zu entwickelnde Methodik der Merkmalselektion.

Im Verlauf dieser Arbeit werden nun unterschiedliche Selektionsstrategien mit der Transinformation kombiniert und deren Eigenschaften miteinander verglichen. In diesem Zusammenhang erweist sich die Kombination von Transinformation mit der sogenannten Forward Selection Strategie als besonders interessant. Es wird gezeigt werden, dass diese Kombination die praktische Berechenbarkeit für hochdimensionale Datenräume, im Vergleich zu anderen Vorgehensweisen, tatsächlich erst ermöglicht. Im Anschluss daran wird die Konvergenz dieses neuen Verfahrens zur Merkmalselektion bewiesen. Wir werden weiterhin sehen, dass die erzielten Ergebnisse bemerkenswert nahe an der optimalen Lösung liegen und im Vergleich mit einer alternativen Selektionsstrategie deutlich überlegen sind.
Parallel zur eigentlichen Selektion der relevanten Merkmale ist es mit der in dieser Arbeit entwickelten Methode nun auch problemlos möglich, eine nichtlineare Analyse der zeitlichen Abhängigkeiten von ausgewählten Merkmalen durchzuführen. Eine anschließende dynamische Modellierung kann somit wesentlich effizienter durchgeführt werden, da die entwickelte Merkmalselektion zusätzliche Information hinsichtlich des dynamischen Zusammenhangs von Eingangs- und Ausgangsdaten liefert. Mit der in dieser Arbeit entwickelten Methode ist nun letztendlich gelungen was vorher nicht möglich war. Das quantitative Erfassen der nichtlinearen Zusammenhänge zwischen dedizierten Sensorsignalen, um diese in eine effiziente Merkmalselektion einfließen zu lassen.
Im Gegensatz zur *Merkmalsextraktion*, hat die in diese Arbeit entwickelte Methode der nichtlinearen *Merkmalselektion* einen weiteren entscheidenden Vorteil. Insbesondere bei sehr kostenintensiven Messungen können diejenigen Variablen ausgewählt werden, die hinsichtlich der Abbildung auf eine Ausgangsgröße den höchsten Informationsgehalt tragen. Neben dem rein technischen Aspekt, die Selektionsentscheidung direkt auf den Informationsgehalt der verfügbaren Daten zu stützen, kann die entwickelte Methode ebenfalls im Vorfeld kostenrelevanter Entscheidungen herangezogen werden, um Redundanz und die damit verbundenen höheren Kosten von vornherein gezielt zu vermeiden.

Eine Übersicht der gängigen Techniken zur Merkmalsextraktion und Merkmalselektion wird in Kapitel 3 gegeben. Die Zusammenstellung beinhaltet neben den Standardmethoden wie *Principal Component Analysis*, *Factor Analysis* und der *Independent Component Analysis* auch eine Reihe von nichtlinearen Techniken der Datenanalyse. Die sog. *Entscheidungsmodelle* und deren Algorithmen werden in einem eigenen Abschnitt vorgestellt. Im weiteren Verlauf, wird schließlich auf einige dieser Standardmethoden Bezug genommen. Hierbei wird ein Vergleich mit der vorgeschlagenen GMI basierten Methodik durchgeführt, dessen Ergebnisse anschließend dargestellt und diskutiert werden.

Der in dieser Arbeit eingesetzte Modellierungsansatz basiert auf neuronalen Netzen als datengetriebene, nichtlineare Approximationsmethode. Ein detaillierter Überblick über diesen universell anwendbaren, neuronalen Modellierungsansatz wird in Kapitel 4 gegeben. Dieser beinhaltet sowohl die Komponenten neuronaler Netze, als auch verschiedene Netzwerktopologien und Trainingsalgorithmen. Es werden einige, zu einem späteren Zeitpunkt, verwendete Trainingsalgorithmen vorgestellt, um den Zusammenhang zwischen der Komplexität der verwendeten Netzwerktopologie und dem damit verbundenen, überproportional steigenden Berechnungsaufwand hervorzuheben. Da der Berechnungsaufwand mit der Dimension des Eingangsvektors in unmittelbarem Zusammenhang steht, motiviert dieser nun seinerseits die Notwendigkeit einer gezielten Dimensionsreduktion, um die praktische Berechenbarkeit der resultierenden neuronalen Netze im realen Feldeinsatz gewährleisten zu können.

Der theoretische Hintergrund der Rényi-Entropiemaße und des daraus abgeleiteten Begriffes einer verallgemeinerten Transinformation wird in Kapitel 5 ausführlich beschrieben. Nach der Einführung der verallgemeinerte Transinformation wird eine Matrixschreibweise zur effizienten Berechnung derseben vorgestellt. Im Anschluss daran, wird die algorithmische Komplexität der vorgeschlagenen Methode ermittelt und einige Aspekte der Implementierung auf modernen Rechnersystemen betrachtet. Um das Leistungsvermögen und die Einsatzmöglichkeiten der Transinformation isoliert zu untersuchen, erfolgt zunächst die Analyse eines nichtlinearen, dynamischen Prozesses. Hierbei wird die Fähigkeit der Transinformation demonstriert, implizite strukturelle- und zeitliche Abhängigkeiten einzig aufgrund der beobachteten Prozessdaten zu identizifieren. In einem weit-

eren Schritt werden diese Daten nun mit einem Anteil von fehlenden Werten versehen und analysiert, um das Verhalten der Transinformation hinsichtlich sogenannter *missing values* zu untersuchen.

Kapitel 6 behandelt die theoretischen Grundlagen der optimalen Merkmalselektion. Vor diesem Hintergrund werden verschiedene Selektionsstrategien in Verbindung mit der verallgemeinerten Transinformation untersucht. Von besonderem Interesse erweist sich in diesem Kontext die Forward Selection Strategie, deren monotone Konvergenz anschließend gezeigt werden wird. In einem separaten Abschnitt wird die Anwendung der Transinformation um die zeitliche Komponente erweitert und deren Berechenbarkeit anhand von realen Messdaten überprüft und diskutiert.

Der Nachweis für die praktische Anwendbarkeit der verallgemeinerten Transinformation zur Merkmalselektion aus Realdaten erfolgt schließlich in Kapitel 7. Um das breite Spektrum der Einsatzmöglichkeiten zu verdeutlichen, wird die vorgeschlagene Methode zur Lösung von Aufgabenstellungen aus völlig unterschiedlichen technischen Bereichen angewandt.
In der ersten Anwendungsstudie wird die Transinformation zur Analyse von Prozessdaten der Glasherstellung verwendet. Das Ziel hierbei ist es, aus dem vorhandenen Datenbestand diejenigen Prozessvariablen auszuwählen, die für eine anschließende Identifikation mit neuronalen Netzen den höchsten Informationsgehalt aufweisen.
Der zweite Anwendungsfall beinhaltet eine sehr komplexe Aufgabenstellung aus der Automobiltechnik. Hierbei steht der Aufbau einer echtzeitfähigen Verbrennungsregelung für einen Otto-Motor im Fokus der Entwicklung. Der zentrale Punkt einer Verbrennungsregelung ist die neuronale Signalauswertung des Zylinderdruckverlaufes, um die Effizienz der einzelnen Verbrennungsvorgänge bestimmen zu können. Die Transinformation wird hierbei erfolgreich eingesetzt, um die Stützstellen der Zylinderdruckkurve festzustellen, die den größten Informationsgehalt hinsichtlich der Bestimmung eines vorgegebenen Motor-Effizienzkriteriums beinhalten.

In Kapitel 8 werden schließlich die Hauptpunkte der Arbeit in komprimierter Form zusammengefasst und Schlussfolgerungen aus den erhaltenen Ergebnissen gezogen. Kapitel 8 schließt mit einigen kritischen Reflexionen, über die in dieser Arbeit entwickelte Methodik, und gibt Ausblicke auf mögliche Weiterentwicklungen der gewonnenen Ergebnisse.

2 Introduction and Overview

2.1 Introduction

In the context of information theory, the term Mutual Information has first been formulated by Claude Elwood Shannon. Information theory is the consistent mathematical description of technical communication systems. To this day, it is the basis of numerous applications in modern communications engineering and yet became indispensable in this field.

This work is concerned with the development of a concept for nonlinear feature selection from scalar, multivariate data on the basis of the mutual information. From the viewpoint of modelling, the successful construction of a realistic model depends highly on the quality of the employed data. In the ideal case, high quality data simply consists of the relevant features for deriving the model. In this context, it is important to possess a suitable method for measuring the degree of the, mostly nonlinear, dependencies between input- and output variables. By means of such a measure, the relevant features could be specifically selected. During the course of this work, it will become evident that the mutual information is a valuable and feasible measure for this task and hence the method of choice for practical applications.

Basically and without the claim of being exhaustive, there are two possible constellations that recommend the application of feature selection. On the one hand, feature selection plays an important role, if the computability of a derived system model cannot be guaranteed, due to a multitude of available features. On the other hand, the existence of very few data points with a significant number of features also recommends the employment of feature selection. The latter constellation is closely related to the so called "Curse of Dimensionality". The actual statement behind this is the necessity to reduce the dimensionality to obtain an adequate coverage of the data space. In other word, it is important to reduce the dimensionality of

the data, since the coverage of the data space exponentially decreases, for a constant number of data points, with the dimensionality of the available data. In the context of mapping between input- and output space, this goal is ideally reached by selecting only the relevant features from the available data set.

The basic idea for this work has its origin in the rather practical field of automotive engineering. It was motivated by the goals of a complex research project in which the nonlinear, dynamic dependencies among a multitude of sensor signals should be identified. The final goal of such activities was to derive so called virtual sensors from identified dependencies among the installed automotive sensors. The difference between virtual- and real sensors is the fact that virtual sensors can be adapted to specific measurement-technology tasks. The particular task consisted in the efficiency optimization for the computation of an essential variable of motor-management. This enables the real-time computability of the required variable without the expenses of additional hardware. The prospect of doing without additional computing hardware is a strong motive force in particular in automotive engineering. In this context, the major problem was to find a feasible method to capture the linear- as well as the nonlinear dependencies.

As mentioned before, the goal of this work is the development of a flexibly applicable system for nonlinear feature selection. The important point here is to guarantee the practicable computability of the developed method even for high dimensional data spaces, which are rather realistic in technical environments.
For specifically accomplishing this task, I combine and suitably integrate a creditable method of information theory with various selection strategies. The employed measure for the feature selection process is based on the sophisticated concept of mutual information. The property of the mutual information, regarding its high sensitivity and specificity to linear- *and* nonlinear statistical dependencies, makes it the method of choice for the development of a highly flexible, nonlinear feature selection framework.

In the course of this work, various selection strategies are combined with the mutual information, its properties are assessed and discussed. In this context, the combination of the so called forward selection strategy with the mutual information turns out to be of particular interest to meet the goals of this work. It will be shown that this combination allows for the practicable computability for high dimensional data spaces in the first place. In a

next step, the monotone convergence of this feature selection algorithm is proved. As a matter of fact, it will be shown that the obtained results are remarkably close to the optimal solution and deliver better results than an alternative selection strategy.
In addition to the mere selection of relevant features, the developed framework is also applicable for the nonlinear analysis of the temporal influences of the selected features. Hence, a subsequent dynamic modelling can be performed more efficiently, since the proposed feature selection algorithm additionally provides information about the temporal dependencies between input- and output variables.
In contrast to feature extraction techniques, the developed feature selection algorithm in this work has another considerable advantage. In the case of cost intensive measurements, the variables with the highest information content can be selected in a prior feasibility study. Hence, the developed method can also be employed to avoid redundance in the acquired data and thus prevent for additional costs.

2.2 Overview

A detailed overview of the state-of-the-art techniques in feature extraction and feature selection is presented in Chapter 3. Apart from standard methods like the *Principal Component Analysis*, the *Factor Analysis* and the *Independent Component Analysis*, this compilation contains also a considerable number of nonlinear data analysis techniques. Finally, the classification models, which are also known as decision models or decision trees, are presented in a separate section.

Since neural networks have been chosen as a data-driven, nonlinear modelling approach for the proof-of-concept in this work, Chapter 4 intents to give a thorough overview of this matter. This includes the network components, as well as different network topologies and neural training algorithms.
The applied algorithms, like Real Time Recurrent Learning or the sophisticated Extended Kalman Filter, will be introduced to demonstrate the correlation between the complexity of the employed network topology and the hence tremendously increasing computational complexity. This increase in computational expense, also motivates the necessity for reducing the input dimension to maintain the practical computability of the networks for real world applications.

The theoretical background of the employed family of Rényi-Entropies and the derived generalized mutual information is introduced in Chapter 5. After the introduction of a well defined measure for the generalized mutual information, an efficient matrix calculus and its implementation aspects on modern computers systems are discussed. Besides the proof of the properties of this sophisticated information measure, an efficient estimation procedure is derived, which is particularly tailored to the presented matrix calculus. The algorithmic complexity class of the proposed method is evaluated in a separate section. Finally, the GMI is employed for the analysis of the implicit structural- and temporal dependencies in nonlinear dynamic process data. Finally, the performance of the GMI with respect to the presence of missing values is also investigated in this chapter.

The theoretical aspects of backward feature elimination, based on the idea of Markov blankets, are addressed in Chapter 6. The quintessence of this theoretical reflection states that once a feature has been considered irrelevant, it can be removed and does not need to be reconsidered again. In the opposite case of forward feature selection, a proof will be developed to show the monoton convergence of this feature selection algorithm. The monoton convergence property justifies the formulation of the proposed feature selection strategy as so called greedy algorithm.
The new point in this context is the employment of Shannon's model of an abstract communication channel for the direct measurement of information as a physical quantity. Different from the application of PCA, ICA or regression models, the GMI operates directly on the raw data without the necessity of assuming any a priori dependence between input- and output. With the background above, the facts from earlier chapters are brought together to derive a constructive and practicable new method for optimal feature selection. In a separate section, the applicability of the mutual information is extended to the identification of time delays. The feasibility of this approach is assessed on the basis of real measurement data.

The proof-of-concept, for the employment of the general mutual information for feature selection, is finally given in Chapter 7. In order to demonstrate the wide applicability of this method, it is used to solve complex tasks in utterly distinct technical disciplines.
In the first application case study, the mutual information is employed for the analysis of glass melting process data. The goal is to select the process variables with the highest information content for a successive neural system identification.

The second application case study is concerned with a rather complex task in the field of automotive engineering. In this context, the goal is to construct a real time capable closed loop combustion controller for a spark-ignition engine. The goal of combustion control is the neural signal processing of the in-cylinder pressure curve to determine the efficiency of the individual combustions. In this case, the general mutual information is applied to determine the sample points of the in-cylinder pressure curve, which deliver the highest information content with respect to the determination of a predefined combustion-efficiency criterion.

In Chapter 8, the objectives of this work are presented in a condensed form and conclusions are drawn from the obtained results. Chapter 8 closes with some critical reflections about the developed methods while giving an inspiration for further research work.

3 State-of-the-Art Feature Extraction and Selection

The tremendous advances in data collection and storage capability in the past decade led to an information overload in almost every scientific discipline. Research engineers working in diverse fields faced the hard fact that, due to the advances in data acquisition, simulation- and internet technology, not only the number of observations but also the dimensionality of the collected data rapidly increased over the past years and yet continues to evolve.

In the context of signal processing, the dimension of the data is the number of variables that are measured with each observation. The problem when handling high-dimensional data sets is, that not all variables are equally important with respect to the underlying process e.g. the combustion process in automotive engineering or the industrial glass production process, as we will discus later in this work.

In this context an important issue has to be mentioned, the so called *curse of dimensionality* [Bis99]. The term *curse of dimensionality* was first verbalized by BELLMAN [Bel61] and refers to the exponential growth of hypervolume as a function of the input dimensionality.
Neural networks can be regarded as nonlinear mappings between an input- and the corresponding output space. Thus, loosely speaking, a neural network needs to cover or represent every part of its input space equally in order to determine how this part should be mapped properly to the output space. Covering the input space takes resources, i.e. the weights which are the adjustable parameters of the neural network. The resources required to maintain an equal coverage of the entire input space grows exponentially with its dimension.
A partial remedy for this problem is to preprocess the input in the right way, for example by selecting the input components according to their de-

gree of statistical independence. Alternatively, the input variables could be chosen due to their importance with respect to the underlying implicit mapping function.
This a-priori information implies a quantitative measure of information content. It is obtained by the so called general mutual information or GMI. This method can help to cope with the curse of dimensionality and will be introduced separately in Chapter 5.
Certain computationally very expensive methods like adaptive algorithms, including neural networks and genetic algorithms, are capable of constructing accurate models from high-dimensional data. However, it is still of interest in many technical applications, especially in real-time environments, to reduce the dimension of the original data prior to any modelling activity.

The fundamental assumption to justify the dimension reduction is that the measured data actually describes a, mostly nonlinear, manifold of lower dimension of the original data space. The goal of dimension reduction is to find a representation of that manifold, i.e. an adequate coordinate system, that will allow to map the data vectors on it and obtain a low-dimensional, compact representation of the original data.

In mathematical terms, the problem of dimension reduction can be stated as follows. Given a p-dimensional random variable $\vec{\xi} = (\xi_1, ..., \xi_p)^T$, find a random variable $\vec{\pi} = (\pi_1, ..., \pi_k)^T$, with $k \leq p$. The new multivariate random variable $\vec{\pi}$ is a lower dimensional representation of $\vec{\xi}$ and does capture the content of the original data. The elements of $\vec{\pi}$ are sometimes also called the *hidden components*. In statistics, the components of such a multivariate random vector are mainly called *variables*. In other fields like in computer science and machine learning they are referred to as *attributes* or *features*.

This chapter is intended to provide a survey of traditional- and current state-of-the-art dimension reduction techniques. This includes the introduction into principal component analysis, the so called non-linear principal component analysis, factor analysis and of course the independent component analysis. At the end of this chapter, further non-linear dimension reduction methods are also referenced.

In general, there exist two major types of dimension reduction methods, linear and non-linear [Fod02]. Characteristic for linear techniques is that the components of the previously mentioned new random variable $\vec{\pi}$, are in

fact defined to be a linear combination of the components of the random variable $\vec{\xi}$:

$$\pi_i = w_{i,1}\xi_1 + ... + w_{i,p}\xi_p, \quad i = 1, ..., k \tag{3.1}$$

or in matrix notation:

$$\begin{pmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_p \end{pmatrix} = \begin{pmatrix} w_{1,1} & ... & w_{1,p} \\ \vdots & \ddots & \vdots \\ w_{k,1} & ... & w_{k,p} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_p \end{pmatrix}, \tag{3.2}$$

where W is the weight matrix of the linear transformation.

Throughout this chapter, it will be assumed that we have n observations in total. Each observation is regarded as a realization of the p-dimensional random variable $\vec{\xi}$ with mean $\vec{\mu} = E[\vec{\xi}] = (\mu_1, ..., \mu_p)^T$ and the $(p \times p)$-covariance matrix $\Sigma = E[(\vec{\xi} - \vec{\mu})(\vec{\xi} - \vec{\mu})^T]$.
The observations of $\vec{\xi}$ are grouped by the so called observation matrix:

$$X = \begin{pmatrix} x_{1,1} & ... & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{p,1} & ... & x_{p,n} \end{pmatrix}, \tag{3.3}$$

where the columns of X represent the p-dimensional realizations of $\vec{\xi}$. Furthermore, it is assumed that the rows of X consists of standardized data with zero mean and standard deviation one. This is a prerequisite for the application of dimension reduction techniques like the principal component analysis.

In terms of the $(p \times n)$ observation matrix X, the transformation can now be formulated as:

$$s_{i,j} = w_{i,1}x_{1,j} + ... + w_{i,p}x_{p,j}, \qquad i = 1, ..., k, \quad j = 1, ..., n \tag{3.4}$$

or in matrix notation:

$$\begin{pmatrix} s_{1,1} & ... & s_{1,n} \\ \vdots & \ddots & \vdots \\ s_{k,1} & ... & s_{k,n} \end{pmatrix} = \begin{pmatrix} w_{1,1} & ... & w_{1,p} \\ \vdots & \ddots & \vdots \\ w_{k,1} & ... & w_{k,p} \end{pmatrix} \begin{pmatrix} x_{1,1} & ... & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{p,1} & ... & x_{p,n} \end{pmatrix}, \tag{3.5}$$

where the $s_{i,j}$ represent the j-th realization of the random variable π_i and $x_{i,j}$ represent the j-th realization of ξ_i.

There are numerous articles and books in statistical literature on various techniques for the analysis of multivariate data sets. A profound introduction into this interesting topic can be found in [MKB95], [Rip96], [Jol86] and [Lee01].

In Section 3.1 and 3.3 the principal component- and the factor analysis are reviewed. They represent the two most widely used linear dimension reduction methods based on second-order statistics. In the case of normal distributed random variables, the previously introduced covariance matrix contains all necessary information about the data. Second-order dimension reduction techniques are fairly simple to implement, as they require standard methods and matrix calculus from linear algebra.
However, many data sets are not realizations of the Gaussian distribution. In this case higher-order linear methods, employing information not contained in the covariance matrix, are appropriate to be used. Such a method, is the independent component analysis. This powerful, linear dimension reduction technique which is based on higher order statistical information, is reviewed in Section 3.4. In Section 3.2, the so called non-linear principal component analysis, which can be considered as a special case of the independent component analysis is also presented. It uses non-linear objective functions to determine the optimal weight matrix. However, the resulting variables are still linear combinations of the original variables. Finally, further non-linear dimension reduction techniques are shortly presented in Section 3.5.

3.1 Principal Component Analysis

The principal component analysis (PCA) is the best linear dimension reduction method in the mean-square error sense [Jac91], [Jol86]. It is a second order method, since it is based on the covariance matrix of the variables. This method is also known as the singular value decomposition method (SVD), the Karkunen-Loève-Transform or the empirical orthogonal function (EOF).
In essence, principal component analysis seeks to reduce the dimension of the input space by finding linear combinations, i.e. the principal components, of the original variables as depicted in Equation 3.2 and 3.4. The first principal component s_1 is the linear combination of the variables $\xi_i,\ i = 1, ..., p$ with the largest variance. The second principal component found is orthogonal to the first one and has the second largest variance, and so on. Since there are as many principal components as the number of original variables, principal component analysis results in a set of orthogonal and thus uncorrelated input variables. To reduce the dimension of the original input data, only those principal components are kept that explain most of the total variance of the data set.
As the empirical variance depends on the scale of the variables, it is important to standardize each variable to have zero mean and standard deviation one. After standardization, the original variables with possibly different scale units are all comparable as mentioned in the definition of the observation matrix X in Equation 3.5. Assuming standardized input data with the empirical covariance matrix

$$\Sigma = \frac{1}{n} X X^T, \tag{3.6}$$

the spectral decomposition theorem can be used to write Σ as

$$\Sigma = U \Lambda U^T, \tag{3.7}$$

where $\Lambda = diag(\lambda_1, ..., \lambda_p)$ is a diagonal matrix containing the ordered eigenvalues $\lambda_1 \leq ... \leq \lambda_p$ of the covariance matrix Σ. The $(p \times p)$-matrix U contains the according eigenvectors.
It is shown in [MKB95], that the desired principal components are obtained as the p rows of the $(p \times n)$-matrix S, where

$$S = U^T X. \tag{3.8}$$

When comparing Equation 3.8 with Equation 3.5, it can be seen that the linear transformation matrix W is equivalent to U^T. It is also shown

in [MKB95], that the subspace spanned by the k eigenvectors has the smallest mean square deviation from X among all subspaces of dimension k.

Another property of the eigenvalue decomposition is that the total variation is equal to the sum of the eigenvalues of the covariance matrix

$$\sum_{i=1}^{p} Var(PC_i) = \sum_{i=1}^{p} \lambda_i \tag{3.9}$$

and that

$$\frac{1}{trace(\Sigma)} \sum_{i=1}^{k} \lambda_i \tag{3.10}$$

gives the cumulative proportion of the variance explained by the first k principal components. By plotting the cumulative proportions in Equation 3.10 as a function of k, one can select the appropriate number of principal components to keep, which explain a certain percentage of the overall variation.

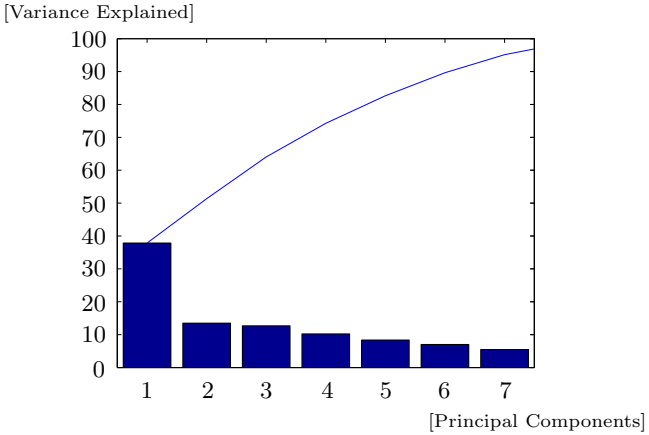


Figure 3.1: *Scree plot of percent variability explained by each principal component*

In statistical literature, such diagrams are called *scree plots*. An exemplary scree plot in depicted in Figure 3.1. However, the number of principal components to be kept can also be determined by defining a fixed threshold λ_0. Only those eigenvectors with a corresponding eigenvalue greater than λ_0 are selected to be kept.

The interpretation of the obtained principal components can be difficult at times. Although the results are uncorrelated variables, which are defined to be linear combinations of the original variables, they do not necessarily correspond to meaningful physical quantities. In some cases, this lack of interpretability, due to the linear transformation of the original variables, is unacceptable to the systems engineer. Anyway, since the principal component analysis provides linear transformations of the original data, this method does not meet the requirements of a nonlinear analysis method. In Section 6.2.3, a sophisticated feature selection method will be suggested, which naturally preserves the interpretability of the original variables. The required theoretical background for this will be thoroughly presented in Chapter 5.

3.2 Non-linear Principal Component Analysis

Nonlinear principal component analysis does introduce nonlinearity in the objective function, but the resulting components are still linear combinations of the original variables. This method can also be seen as a special case of independent component analysis in Section 3.4. As mentioned in [KPO98], there exist various formulations of the non-linear PCA.
A non-linear PCA criterion for the random data vector $\vec{x} = (x_1, ..., x_p)^T$ searches for the components $\vec{s} = (s_1, ..., s_p)^T$ in the form of $\vec{s} = W^T\vec{x}$ by minimizing

$$J(W) = E[\mid \vec{x} - Wg(W^T\vec{x}) \mid^2], \tag{3.11}$$

with respect to the $p \times p$ weight matrix W, where $g(\vec{y})$ denotes the component-wise application of the non-linear function g to the elements of the vector $\vec{y}$. Common choices for the non-linear function g are odd functions like $g(\vec{y}) = \tanh(\vec{y})$ and $g(\vec{y}) = \vec{y}^3$. The optimization in Equation 3.11 can be carried out either by stochastic gradient descent algorithms with the learning parameter c or by an approximate recursive least square algorithm. The recursive least square algorithm converges much

faster than the corresponding gradient descent method with a good final accuracy but a higher computational expense. However, before applying the mentioned algorithms, the data needs to be pre-whitened. Although the nonlinear principal component analysis introduces a nonlinearity in its objective function, this method does not meet the requirements for a nonlinear data analysis method.

3.3 Factor Analysis

The presentation of factor analysis in this section follows the description in [MKB95]. Factor analysis is, like the principal component analysis in Section 3.1, also a linear method based on second-order statistical information. First suggested by psychologists, factor analysis assumes that the measured variables are dependent on some unknown, and often unmeasurable, common factors.
A typical example for the application of factor analysis are random variables defined as various test scores of individuals. Such scores are assumed to be related through a common "intelligence" factor. The goal of factor analysis is to unveil such relations, which can then be used further to reduce the dimension of the data set, following the factor model.

The zero-mean p-dimensional random vector $\vec{x}$ with the according covariance matrix Σ satisfies the so called k-factor model if

$$\vec{x} = \Lambda\vec{f} + \vec{u} \tag{3.12}$$

holds, where Λ is a $(p \times k)$-matrix of constants, $\vec{f}$ is a $(k \times 1)$-vector describing the random factors and $\vec{u}$ is a $(p \times 1)$-vector describing the specific factors. Furthermore, the factors are all uncorrelated and the random common factors are standardized to have zero mean and variance one.

$$\begin{aligned} E(\vec{u}) = 0,\ \ Cov(u_i, u_j) &= 0, \quad i \neq 1, \\ Cov(\vec{f}, \vec{u}\,) &= \mathbf{0}, \\ E(\vec{f}\,) = \mathbf{0},\ \ Var(\vec{f}\,) &= \mathbf{I}, \end{aligned} \tag{3.13}$$

Under these assumptions, the diagonal covariance matrix of $\vec{u}$ can also be written as $Cov(\vec{u}) = \Psi = diag(\psi_{1,1},\ ...,\ \psi_{p,p})$.

If the data covariance matrix Σ can be decomposed as

$$\Sigma = \Lambda\Lambda^T + \Psi, \tag{3.14}$$

it can be shown that the previously mentioned k-factor model holds. Since x_i can be written as

$$x_i = \sum_{j=1}^{k} \lambda_{i,j} f_j + u_i, \qquad i = 1, \ ..., \ p, \tag{3.15}$$

its variance may be decomposed and written as

$$\sigma_{i,i} = \sum_{j=1}^{k} \lambda_{i,j}^2 + \psi_{i,i} = h_i^2 + \psi_{i,i}, \tag{3.16}$$

where the first part h_i^2 is called the *communality* and represents the variance of x_i common to all variables. The second part $\psi_{i,i}$ is called the specific- or unique variance and expresses the contribution in the variability of x_i due to its specific part u_i, which is not shared by other variables. The term $\lambda_{i,j}^2$ measures the magnitude of the dependence of x_i on the common factor f_j. If several variables x_i have high *loadings* $\lambda_{i,j}$ on a given factor f_j, the implication is that those variables measure the same unobservable quantity. Those variables are therefore considered to be redundant.
Unlike the previously presented principal component analysis, the factor model does not depend on the scale of the variables. It is scale-invariant and also holds for orthogonal rotations of the factors.

Two instances of factor analysis are used to derive estimates for the model parameters Λ and Ψ in Equations 3.12 and 3.13. The two methods are known as *principal factor analysis* and *maximum likelihood factor analysis*. Both are described in detail by MARDIA [MKB95].
Since factor analysis is based on a linear model assumption, this methods are again not suitable for nonlinear data analysis.

3.4 Independent Component Analysis

The independent component analysis is a powerful state-of-the-art data analysis method that incorporates higher-order statistical information [Hyv99]. It seeks linear projections, not necessarily orthogonal to each other, that are as close to statistical independence as possible. Since statistical independence is a much stronger requirement than uncorrelatedness, it involves not only second-order but all higher-order statistical information. Formally, the random variables $(\xi_1, \ ..., \ \xi_p)^T$ are mutually uncorrelated if

the following holds:

$$Cov(\xi_i, \xi_j) = E[(\xi_i - \mu_i)(\xi_j - \mu_j)] = 0, \quad 1 \leq i, j \leq p, \quad i \neq j. \tag{3.17}$$

In contrast to uncorrelatedness, statistical independence requires the compound probability density function $f(\xi_1 \ ..., \ \xi_p)$ to factorize into a product of p terms so it can be written as

$$f(\xi_1 \ ..., \ \xi_p) = f(\xi_1) \cdot ... \cdot f(\xi_p). \tag{3.18}$$

For non-gaussian distributed random variables independence always implies uncorrelatedness, but naturally not vice versa.

3.4.1 Motivation of the ICA

In order to provide the motivation that lies behind the independent component analysis, the following example is presented. Imagine you are in a room where two people are speaking simultaneously. Two arbitrarily distributed microphones with different orientations are recording two time dependent signals which are denoted by $x_1(t)$ and $x_2(t)$. Each of these recorded signals is a weighted sum of the original speech signals $s_1(t)$ and $s_2(t)$ emitted by the two speakers. This situation could be expressed by the following system of linear equations

$$x_1(t) = a_{1,1} \ s_1(t) + a_{1,2} \ s_2(t) \tag{3.19}$$

$$x_2(t) = a_{2,1} \ s_1(t) + a_{2,2} \ s_2(t), \tag{3.20}$$

where $a_{1,1}$, $a_{1,2}$, $a_{2,1}$ and $a_{2,2}$ are some parameters that depend on the distance of the microphones from the speakers, the attenuation of the original signals and of the acoustic properties of the room itself.
It would now be very useful, if one could estimate the two original speech signals $s_1(t)$ and $s_2(t)$, using only the recorded signals $x_1(t)$ and $x_2(t)$. This is commonly known as the *cocktail party problem*. Since no information about the source signals $s_1(t)$ and $s_2(t)$ is available, the technique for the solution of the cocktail party problem is called *blind source separation*.

Actually, if the parameters $a_{i,j}$ were all known, Equation 3.19 and 3.20 could be easily solved by classical methods of linear algebra. However, the point is that neither the parameters $a_{i,j}$ nor the source signals $s_i(t)$ are known by the time, the signals $x_1(t)$ and $x_2(t)$ are recorded. This is what makes the problem considerably more difficult, since the missing source

signals prevent us from accomplishing a standard linear system inversion.

The formal definition of independent component analysis employs the statistical "latent variables" model [JH91] [Com94]. This assumes that at least p linear mixtures $x_1, ..., x_p$ of p independent components $s_1, ..., s_p$ are observed. In Equation 3.21, the time indices t are omitted, since each mixture x_j as well as each independent component s_i now represents a random variable, instead of a proper time signal. The value $x_j(t)$, e.g. a microphone signal in our cocktail party problem, is a sample of this particular random variable at a specific time.

$$x_j = a_{j,1}\ s_1 + a_{j,2}\ s_2 + ... + a_{j,p}\ s_p, \quad j = 1, ..., m \tag{3.21}$$

Since vector-matrix notation is much more convenient, the above mixing model can also be written in the form

$$\vec{x} = A\ \vec{s}. \tag{3.22}$$

The statistical model in Equation 3.22 is called *independent component analysis model* or in short: *ICA model*. The ICA model is a generative model which describes how the observed data x_j is generated by the process of mixing the components s_i. In this context, the independent components s_i are also called latent variables, since they cannot be observed directly.
In this context, it should be mentioned again that the independent component analysis is based on a linear model assumption. Regardless of its amazing properties, this method is not sensitive to nonlinear dependencies and cannot be employed for nonlinear data analysis.
Due to the fact that ICA is a rather new research topic, there exist various definitions in literature. Perhaps the most general definition of ICA can be given as follows.

Definition 3.1
Independent component analysis (ICA) of the random vector $\vec{x}$ consists of finding a linear transform $\vec{s} = W\vec{x}$ so that the components s_i are as independent as possible, in the sense of optimizing some function $F(s_1, ..., s_p)$ that measures the degree statistical independence.

The identifiability of the ICA model in Equation 3.22 has been treated extensively by COMON [Com94]. By imposing the following fundamental restrictions, in addition to the basic assumption of statistical independence, the identifiability of the model can be assured.

1. All independent components s_i must have non-gaussian distribution.

2. The number of observed linear mixtures m must be at least as large as the number of independent components p.

3. The parameter matrix A must be of full column rank.

Usually, it is also assumed that the realizations of $\vec{x}$ and $\vec{s}$ in Equation 3.22 are centered, which is no restriction, since this can always be accomplished by subtracting the mean of the particular random variable from the according data samples.

The first restriction in the above list, referring to non-gaussian distributions, is necessary for the identifiability of the ICA model [Com94]. For gaussian random variables, the mere uncorrelatedness already implies statistical independence.
The second restriction, that $m \geq p$ must hold to assure the identifiability of the ICA model, may be relaxed in cases where only the mixing matrix A is of interest. Even if $m < p$, the mixing matrix A seems to be identifiable [Car91]. The drawback here is the unavailability of the independent components due to the noninvertibility of A. Since most of existing theory for independent component analysis is not valid in this case, the second assumption is usually made for practical applications. Recent results for the case $m < p$, which is also called ICA with overcomplete basis, can be found in [HCO99], [LS98b],[LS98a].
While the second restriction assures the existence of the mixing matrix A, the third restriction implies the existence of its inverse or at least its pseudo-inverse.

A rather insignificant indeterminacy of the ICA model is that the columns of A and thus the independent components s_i can only be estimated up to a multiplicative constant. This is the case, because any constant multiplying an independent component in Equation 3.22 could be canceled by dividing the corresponding column of the mixing matrix A by the same constant. In this context, it is convenient to predefine the independent components s_i to have unit variance. This makes the independent components unique, up to an algebraic sign which could be different for each component s_i.

As mentioned above, independent component analysis is very closely related to the method called *blind source separation* or *blind signal separation*. The term "source" refers to an independent component, which is equivalent to the original signal, e.g. the speaker in the previously described

cocktail party problem. "Blind" means that very little to nothing is known about the mixing matrix and the according source signals.
When employing ICA for feature extraction, the random variables s_i are referred to as the features. The elements of the mixing matrix A are representing the contribution of the according features to the observation variables x_j.
In contrast to the principal component analysis described in Section 3.1, the above definition of the ICA does not imply a proper order of the independent components. In the case, when ICA is be used as an instrument for redundancy reduction or feature extraction, the existence of such an order is essential. One way to define such a particular order among the independent components might be the norms of the columns of the mixing matrix, which could be interpreted as the contributions of the independent components s_i to the variance of the observed mixtures x_j. Ordering the components s_i according to the descending norm of the corresponding columns in the mixing matrix A, could establish an ordering similar to that of a principal component analysis. The dimension reduction could then be performed in the way, which is described in Section 3.1.
In many applications however, the presence of measurement noise would be more realistic. As a consequence, this would result in an additional noise term in the ICA model. However, for the sake of simplicity of this overview, any noise terms are omitted in the formulation of the model. Since describing the estimation of a noise-free model is already a challenging task, this approach seems to be sufficient for the mere description of the ICA method.

3.4.2 Estimation of the ICA

The estimation of the ICA model is performed by the minimization or maximization of a so called *objective function*. Often such a function is also called a *contrast function*, a *loss function* or a *cost function* by some authors. Basically, this refers to any function that enables the estimation of the parameters of the ICA model by implementing a measure of statistical independence, as previously mentioned in Definition 3.1.
In this context, it is about to mention that the ICA method is highly dependent on the formulation of the objective function and the according optimization algorithm. In the case of explicitly formulated objective functions, almost any of the classical optimization methods can be used for optimizing the objective function. The statistical properties of the ICA method, e.g. consistency, asymptotic variance and robustness, depend on the choice of this objective function. The algorithmic properties like con-

vergence speed, memory requirements and numerical stability depend on the employed optimization algorithm itself.

The choice of the objective function depends on the intention how the ICA model should be identified. On the one hand, the independent components could successively be estimated one by one. This implies the use of so called *single-unit objective functions*. On the other hand, the data model, i.e. all independent components, could be estimated at the same time. In this case, *multi-unit objective functions* have to be employed for optimization.

Multi-unit objective functions

If the independent components of the ICA model in Equation 3.22 are all to be estimated at the same time, multi-unit objective functions are used. In the following, various commonly used multi-unit objective functions are presented.

1. Maximum Likelihood
   A very popular approach for the solution of the ICA model is the maximum likelihood estimation. For the noise-free data model in Equation 3.22, it is possible to formulate a likelihood function and then estimate the model by a maximum likelihood method [PGJ92]. The log-likelihood function appears to have the following form:

$$L = \sum_{t=1}^{T} \sum_{i=1}^{m} \log\ f_i(\vec{\mathbf{w}}_i^T \mathbf{x}(t)) + T\ \log \mid \det\ \mathbf{W} \mid, \tag{3.23}$$

   where $\mathbf{W} = (\vec{\mathbf{w}}_1, ..., \vec{\mathbf{w}}_m)^T$ denotes the inverse mixing matrix A^{-1}. The probability density functions f_i of the independent components s_i are assumed to be known and $\vec{\mathbf{x}}(t),\ t = 1, ..., T$ are the realizations of the multivariate random vector $\vec{\mathbf{x}}$ which represents the observation data.
   The great advantage of the maximum likelihood approach is its asymptotic efficiency, which is a well-known result from estimation theory [Sch95]. However, there are also some major drawbacks of this approach. First, that probability density functions of the independent components are required prior to estimation. The second drawback is the sensitivity of the maximum likelihood method to outliers, since the probability density functions of the independent components are assumed to have certain shapes. However, robustness against outliers should be an essential property of any estimator.

2. Mutual Information
   Theoretically the most satisfying objective function in the multi-unit case of the ICA is the mutual information. Analog to the definition in Section 5.1, the mutual information I between m *scalar* random variables y_i, $i = 1, ..., m$ is defined as the sum of entropies

$$I(y_1, ..., y_m) = \sum_i H(y_i) - H(\mathbf{y}). \tag{3.24}$$

   The mutual information is a natural measure of dependence between, possibly multivariate, random variables. This fact will be presented in detail, however in another context, in Chapter 5. It is always non-negative, and zero if and only if the variables are statistically independent. Thus the mutual information takes into account the whole dependence structure of the random variables.
   Finding a transformation that minimizes the mutual information between the components s_i is a natural way to estimate the ICA model [Com94] and to abide by Definition 3.1. The drawback of this measure of dependence is the computational expense to assess the mutual information, especially for multivariate random variables.

3. Non-linear cross-correlation
   Since the paper published by JUTTEN [JH91], several authors have used the principle of canceling non-linear cross-correlations to obtain the independent components [JH91], [CHL96]. Such non-linear cross-correlations are of the form $E[g_1(y_i)\ g_2(y_j)]$, where g_1 and g_2 are some suitably chosen odd non-linearities. If y_i and y_j are independent, these cross-correlations are zero – on the condition that y_i and y_j have symmetric probability densities. The objective function may be only formulated implicitly and an explicit formulation may not even exist. In that case, the non-linearities must be chosen according to the probability density functions of the independent components [Lam96], [LWB95], [TJ91].

Single-unit objective functions

The expression *single-unit objective function* is used to indicate any function whose optimization enables the estimation of a single independent component. Instead of estimating the complete ICA model all at once, only one vector w is determined. Thus, the linear combination $w^T x$ corresponds to one independent component s_i. This procedure can be iterated to find more independent components successively.

The use of single-unit objective functions is motivated by the fact, that in many applications it is enough to find some of the independent components. Prior knowledge of the number of independent components is not needed, since they can be estimated one by one. In the following, some examples of single-unit objective functions are presented.

1. Negentropy
   From the viewpoint of information theory, the most natural one-unit objective function is the negative normalized entropy or *negentropy*. Regarding Equation 3.24, one might conclude that the independent components correspond to directions in which the differential entropy of $w_T x$ is minimized. This turns out to be approximately right. However, a modification has to be made, since differential entropy is not invariant for scale transformations. To obtain a linear invariant form, the negentropy J is defined as

   $$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}), \tag{3.25}$$

   where $\mathbf{y}_{gauss}$ is a Gaussian random vector with the same covariance matrix as $\mathbf{y}$. The negentropy J is always non-negative and it is zero if and only if $\mathbf{y}$ has a Gaussian distribution [Com94].

2. General objective functions
   In order to generalize the existing objective functions and to avoid their drawbacks, new single-unit objective functions for ICA have been proposed [Hyv98], [Hyv97]. These objective functions combine good statistical properties with plain algorithmic implementation and require no prior knowledge about the densities of the independent components, in contrast to the maximum likelihood estimation. The generalized objective functions are without exception measures non-normality [Com94]. A family of such measures of non-normality could be constructed by employing any continuous function G and considering the difference between the expectation of G for the actual data and its expectation for Gaussian data. In the following definition, the objective function J measures the non-normality of a zero-mean random variable y employing an even, on-quadratic and sufficiently smooth function G

   $$J_G(y) = \mid E_y[G(y)] - E_\nu[G(\nu)] \mid^p, \tag{3.26}$$

   where ν is a standardized Gaussian random variable, y is assumed to be normalized to unit variance and the exponent is typically $p = 1, 2$.

3.5 Non-linear Methods of Data Analysis

This section is intended to provide some further aspects of the rather wide range of non-linear methods which can also be employed for the purpose of feature extraction or -selection. It should be mentioned, that most methods are not directly intended to serve as a tool for this purpose and need more or less additional modification to suit the imposed requirements.

3.5.1 Non-linear Independent Component Analysis

In contrast to the previously presented independent component analysis, the non-linear ICA replaces the linear mixing matrix A by a non-linear vector-valued function f.

Given a p-dimensional zero-mean and non-Gaussian random variable $\vec{x}$, the non-linear ICA model substitutes the linear transformation in Equation 3.22 by the following:

$$(x_1, ..., x_p)^T = \mathbf{f}(s_1, ..., s_k)^T, \tag{3.27}$$

where $\mathbf{f}$ is a p-dimensional vector-valued function. The problem of identifiability of a general non-linear ICA model makes its estimation rather difficult. Few publications, considering estimation of special cases of non-linear ICA models are presented in [Hyv99]. An overview of the problem, along with a maximum likelihood and a Bayesian ensemble method for estimation can be found in [Kar00].

3.5.2 Multidimensional Scaling

With n data points in a p-dimensional space and an $(n \times n)$-matrix of proximity measures among these points, multidimensional scaling produces a k-dimensional representation of the original data set, with $k \leq p$. The distances between the data points in the k-dimensional space reflect the proximity relations among the original data points in p-dimensional space. In general, a distance measure like the Euclidian distance, the Manhattan- or the Maximum-Norm is employed to model the similarity among the data points.
Multidimensional scaling is typically used to transform higher dimensional data into two- or three-dimensional representations. After transformation, the data can be visualized to unveil hidden structures in the data. A rule of thumb to determine the maximum number of k, is to ensure that there

are at least twice as many pairs of data points as there are parameters to be estimated.

Given the data points $\{\mathbf{x}_i\}_{i=1}^n \in I\!R^p$ and a symmetric distance matrix $\Delta = \{\delta_{i,j}\}, \quad i,j = 1,...,n$, the result of a k-multidimensional scaling will be the set of points $\{\mathbf{y}_i\}_{i=1}^n \in I\!R^k$such that the distances $d_{i,j} = d(\mathbf{y}_i, \mathbf{y}_j)$ are as close as possible to a function f of the corresponding proximities $f(\delta_{i,j})$ in the original data set. A detailed presentation of multidimensional scaling and its application can be found in [CC01] and [MKB95].

3.5.3 Regression Analysis

Regression methods can be used for dimension reduction when the goal is to model a response variable y in terms of a set of variables x_i. The identified regression function can be linear or non-linear. In statistics, the variables x_i are traditionally called the independent or explanatory variables, while y is the response- or dependent variable. In the regression context, it is generally assumed that the variables x_i are carefully selected, uncorrelated and relevant for the explanation of y. However, in current data mining applications those assumptions barely hold. Prior variable-selection or dimension-reduction is therefore needed.

A well-known statistical variable selection method is the so called step-wise regression, where different models are fit using all possible subsets of the explanatory variables. The results are then compared by calculating various quality measures. The subset showing the best quality measure is then chosen. It contains the explanatory variables with the reduced dimension. A similar approach, selecting the most relevant features by evaluating random subsets of the original features, is called the *wrapper method* in the machine learning community [KJ98].

3.5.4 Kohonen's Self-Organizing Feature Maps

Self-organizing neural network models generate mappings from D-dimensional signal spaces to L-dimensional topological structures. These mappings preserve the neighborhood relations in the input data and have the property to represent regions of high signal density on corresponding large parts of the topological structure. This makes them interesting for applications in various disciplines ranging from speech recognition [Koh88] to data compression [Sea91].

One possible application objective is feature selection, i.e. finding a low-dimensional subspace of the input vector space containing most or all of the information of the original input data. Linear subspaces can be computed directly by principal component analysis, as it is demonstrated in Section 3.1. The Kohonen feature map allows the projection onto non-linear, discretely samples subspaces with previously chosen lower dimensionality. A detailed description of this very interesting method and its extensions can be found in [Koh90], [Fri94].

Let $\{\vec{t}_k\}_{k=1}^{K} \in I\!R^D$ be samples of data space and let $\{\vec{\mu}_m\}_{m=1}^{M} \in I\!R^D$ define the randomly initialized reference vectors of the feature map. For a given data sample $\vec{t}_k$, the feature map network identifies the index of the winning neuron i^* by calculating

$$i^* = \arg \min_{m=1,\ldots,M} \mid \vec{t}_k - \vec{\mu}_m \mid_D, \tag{3.28}$$

where $\mid \cdot \mid_D$ typically defines the Euclidean distance in $I\!R^D$.

Instead of updating only the reference vector $\vec{\mu}_{i^*}$ of the winning neuron i^*, all neurons in its neighborhood $j \in N_{i^*}(d)$ of radius d are updated according to

$$\vec{\mu}_j^{\,new} = \vec{\mu}_i^{\,old} + \alpha(\vec{t}_k - \vec{\mu}_j), \tag{3.29}$$

where the neighborhood is defined as $N_{i^*}(d) = \{j \mid dist(i^*, j) < d\}$.
The distance measure $dist(i, j)$ defines the distance between the neurons i and j in the previously mentioned L-dimensional topological structure onto which the mapping is performed. The learning rate α and the radius d of the neighborhood are usually simulation time dependent and decrease with the number n of iterations performed.
After a number of iterations with randomly chosen data vectors $\vec{t}_k$ and convergence, the feature map network depicts the distribution and topology of the input vectors it has been trained on.
Although Kohonen's self-organizing feature maps are useful in many application, they have two drawbacks. The general convergence property of the algorithm depends mainly on the choice of the learning rate α and the neighborhood function $N_i(d)$). Since there is no rule, these parameters have to be set based on empirical values.
Finally, the major drawback is the absence of an explicitly formulated measure to be optimized, like e.g the well defined mutual information in Chapter 5.

3.6 Inducing Classification Models from Data

Classification models, which are also known as decision models or decision trees, are used to divide data objects into a predefined set of categories or classes. Each data object has the same structure, consisting of a number of attributes. One of these attributes represents the so called category of the object.

Attribute	Values
Outlook	sunny, overcast, rain
Humidity	0% - 100%
Windy	true, false
Play	yes, no

Table 3.1: *Structure of a data object including the non-category attributes* `Outlook`, `Humidity`, `Windy` *and the category attribute* `Play` *of a decision model for playing golf.*

Table 3.1 depicts the structure of data records reporting on weather conditions, which might be used as inputs for the decision of playing golf. The categorical attribute `Play` in the last line, specifies whether to play or not, depending on the non-categorical attributes `Outlook`, `Humidity` and `Windy`. Usually, the category attribute only represents binary values like {*yes*, *no*}, {*true*, *false*} or something equivalent. In any case, one of its values will determine a required decision.
The task is now to construct a decision tree that, on the basis of answers to questions about the non-category attributes, correctly predicts the value of the category attribute.

In the eighties and nineties, QUINLAN [Qui86], [Qui93] introduced algorithms to obtain such decision trees from data sets. In this context, the most important algorithms to be mentioned for the deduction of decision trees are *ID3* and *C4.5*. The ID3-Algorithm employs the so called *entropy*, a concept of information theory, as a heuristic function for the construction of decision trees.

The basic conceptual ideas behind the ID3-Algorithm are:

- Each node of the decision tree corresponds to a non-categorical attribute. Each edge of the tree is associated to a possible value of that attribute.
- A leaf specifies the expected value of the categorical attribute, which is described by the path from the root of the tree to this particular leaf.
- Each node of the decision tree should be associated to the most informative non-categorical attribute among all attributes not yet considered in the current path from the root.
- Entropy[1] is used to measure the previously mentioned importance of a non-categorical attribute.

C4.5 is an extension of the ID3-Algorithm. It accounts for missing values, continuous attribute value ranges, the pruning of decision trees and the derivation of rules from the latter.

Outlook	Humidity	Windy	***Play***
sunny	85	false	no
sunny	90	true	no
overcast	78	false	yes
rain	96	false	yes
rain	80	false	yes
rain	70	true	no
overcast	65	true	yes
sunny	95	false	no
sunny	70	false	yes
rain	80	false	yes
sunny	70	true	yes
overcast	90	true	yes
overcast	75	false	yes
rain	80	true	no

Table 3.2: *Training data set the golf example*

Table 3.2 depicts the training set for the golf decision model. Notice that in

[1] Entropy is an important concept in communication theory and has been introduced by SHANNON[Sha48].

this example the non-categorical attribute `Humidity` has continuous range. Since ID3 does not directly handle continuous range attributes, the extensions of the C4.5-Algorithm will be also introduced below.

3.6.1 Basic Definitions

As mentioned above, the entropy is employed to measure the importance of a non-categorical attribute when inducing a decision tree. The basic definitions, required to understand the decision tree algorithms, are given in this section. However, a detailed description of information theory is presented in Chapter 5. Let ξ be a discrete random variable with the probability distribution

$$P(\xi = x_m) = \{p_m\}_{m=1}^{M}, \tag{3.30}$$

where M is the number of possible realizations of the random variable ξ. In the sense of information theory, this random variable characterizes an information source producing realizations of $\xi = x_m$ with probability p_m. The entropy, which measures the information conveyed by this distribution, is an integral measure for the uncertainty about the realizations of ξ. The most commonly known entropy measure is the so called Shannon-Entropy which is defined as

$$H(\xi) = -\sum_{m=1}^{M} p_m \log_2 p_m. \tag{3.31}$$

Note that the more uniform a probability distribution is, the higher is its degree of uncertainty and hence its assumed information content.
For example, let us consider the distribution $P(\xi) = \{0.67, 0.33\}$. According to Equation 3.31, its entropy is $H(\xi) = 0.92$. A uniform distribution $P(\xi) = \{0.5, 0.5\}$ has the entropy $H(\xi) = 1.0$ and thus is assumed to contain a higher amount of information, from the viewpoint of information theory.

If a set T of data records is partitioned into disjoint classes $\{C_1, ..., C_M\}$ on the basis of the value of a categorical attribute, then the information needed to identify the class of an element of T is defined as: $Info(T) = H(\xi)$, where ξ is the random variable characterizing the partition $\{C_1, ..., C_k\}$ with its probability distribution $P(\xi = C_m) = \{\frac{|C_m|}{|T|}\}$.
In the golfing example, we have the distribution $P(\xi) = \{\frac{9}{14}, \frac{5}{14}\}$ and thus $Info(T) = 0.94$.

If the set T is partitioned in advance on the basis of the value of a non-categorical attribute X into sets $\{T_1, ..., T_n\}$, then the information needed to identify the class of an element of T becomes

$$Info(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \; Info(T_i). \tag{3.32}$$

The application of Equation 3.32 regarding the non-categorical attribute `Outlook` in the golf data set in Table 3.2 yields

$$\begin{aligned} Info(\texttt{Outlook}, T) = & \\ & \frac{5}{14} \; H\left(\left\{\frac{3}{5}, \frac{2}{5}\right\}\right) \qquad (\texttt{sunny}) \\ & + \frac{4}{14} \; H\left(\left\{\frac{4}{4}, 0\right\}\right) \qquad (\texttt{overcast}) \\ & + \frac{5}{14} \; H\left(\left\{\frac{2}{5}, \frac{3}{5}\right\}\right) \qquad (\texttt{rain}) \\ & = 0.694. \end{aligned} \tag{3.33}$$

Consider furthermore the quantity $Gain(X,T)$ defined as

$$Gain(X,T) = Info(T) - Info(X,T). \tag{3.34}$$

It represents the difference between the information needed to identify an element of T, and the information needed to identify an element of T after the value of non-categorical attribute X has been obtained. This quantity specifies the information gain due to attribute X.
For the attribute `Outlook` in the golfing example, the information gain is: $Gain(\texttt{Outlook}, T) = 0.94 - 0.694 = 0.246$. If the attribute `Windy` is considered instead, it can be seen that $Info(\texttt{Windy}, T)$ is 0.892 and $Gain(\texttt{Windy}, T)$ is 0.048. Thus, the attribute `Outlook` offers a greater information gain than the attribute `Windy`.
The quantity $Gain(X,T)$ can be used to sort the non-categorical attributes due their information gain. Thus, it can also be used to build decision trees, where each node is associated with the attribute showing the greatest gain among all attributes not yet considered in the path from the root of the tree.

3.6.2 The ID3 Algorithm

Let T be a training data set, similar to the set of records in Table 3.2. Given a categorical attribute C and a set of non-categorical attributes R, the ID3 algorithm is used to build decision trees. In the following, the ID3 algorithm is rendered in pseudo code notation.

Algorithm 3.1 (ID3)

```
function Tree = ID3 (R: set of non-categorical attribute,
                     C: categorical attribute,
                     T: training set)
begin
    If T is empty,
        return a single node with value Failure;

    If all records in T have the same value for
    the categorical attribute,
        return a single node with that value;

    If R is empty,
        return a single node with the value of the most
        frequent categorical attribute values in T;

    Let D be the attribute with highest Gain(D,T)
    among all attributes in R;

    Let {d_j|j = 1,...,m} be the values of attribute D;

    Let {T_j|j = 1,...,m} be subsets of T consisting
    of records with value d_j for attribute D;

    Return a tree with root labeled D and
    edges labeled d_1,...,d_m connecting to the subtrees
    ID3(R - {D}, C, T_1), ..., ID3(R - {D}, C, T_m);
end
```

The algorithm recursively builds a decision tree from a given training data set T. Since the attributes with the highest gain are removed from the recursive calls to the function, the most important attributes appear nearest to the root of the final decision tree.

3.6.3 Information Gain Ratios

The quantity gain in Equation 3.34 tends to favor attributes that have a large number of distinct values. For instance, if an attribute D has distinct value in each record of the data set T, then $Info(D,T)$ becomes zero and thus $Gain(D,T)$ is maximal. To compensate for this, QUINLAN [Qui93] suggested to use the following quantity instead.

$$GainRatio(X,T) = \frac{Gain(X,T)}{SplitInfo(X,T)}, \tag{3.35}$$

where $SplitInfo(X,T)$ is the information due to the split of T on the basis of the value of the attribute X. The quantity $SplitInfo(X,T)$ is defined as

$$SplitInfo(X,T) = \sum_{i=1}^{n} H(T_i), \tag{3.36}$$

where $\{T_1, T_2, ..., T_m\}$ is the partition of the training data set T due to the values of the attribute X.

In the case of the golfing example, the non-categorical attributes `Outlook` and `Windy` yield

$$\begin{aligned} SplitInfo(\texttt{Outlook},T) &= H\left(\left\{\frac{5}{14},\frac{4}{14},\frac{5}{14}\right\}\right) = 1.577, \\ SplitInfo(\texttt{Windy},T) &= H\left(\left\{\frac{6}{14},\frac{8}{14}\right\}\right) = 0.985. \end{aligned} \tag{3.37}$$

(3.38)

Finally, the quantity $GainRatio(\texttt{Outlook},T)$ is 0.156, compared to $Gain(\texttt{Outlook},T) = 0.246$. The gain ratio for the attribute `Windy` is $GainRatio(\texttt{Windy},T) = 0.049$, instead of $Gain(\texttt{Windy},T) = 0.046$.

3.6.4 C4.5 Extensions

Compared to the original ID3 algorithm, C4.5 introduces a number of extensions accounting for the previously mentioned deficiencies of ID3.

- In building a decision tree, C4.5 can deal with training sets containing records with unknown attribute values. This is done by evaluating the required gain, or the gain ratio, for those records in the data set, where that attribute is actually defined.

- In using a decision tree, records with unknown attribute values can also be processed by estimating the probability of the various possible results.
  Considering the decision tree of the golfing example in Figure 3.2, a new record with `Outlook` = `sunny` and unknown value for the attribute `Humidity` is processed as follows:
  Since the value of `Outlook` is known, the edge labeled `sunny` is followed from root node `Outlook` to its child node `Humidity`. At that point, the value of `Humidity` is unknown. However, if `Humidity` $\leq$ 75% it can be observed in the training data set, that there are two records where the categorical attribute `Play` = `yes`. If `Humidity` $>$ 75%, there are three records with `Play` = `no`. Thus the answer for this record, with respect to the question about the probability for playing golf, is 0.4 for "Play" and 0.6 for "Don't Play".

- In the case of continuous range attributes, C4.5 performs a global search to determine the maximum information gain. Let R_i be an attribute of continuous rage and let $A_1, ..., A_m$ be all its values in the training set. For each value A_j with $j = 1, ..., m$ the set of training records T is partitioned into subsets $T_{j,1} = \{t \in T | A_k \leq A_j,\ k = 1, ..., m\}$ and $T_{j,2} = \{t \in T | A_k > A_j,\ k = 1, ..., m\}$. For each of this partitions, the gain or the gain ratio is computed. The partition which maximizes this quantity is finally chosen.
  In the golfing example, the best partition for the attribute `Humidity` is determined to be at 75%. In the decision tree, the range intervals for this attribute then become $]-\infty, 75]$ and $]75, +\infty[$. Notice that this method involves a substantial number of computations, dependent on the number of records in the training data set.

In the case of the golfing example, the following decision tree is finally obtained.

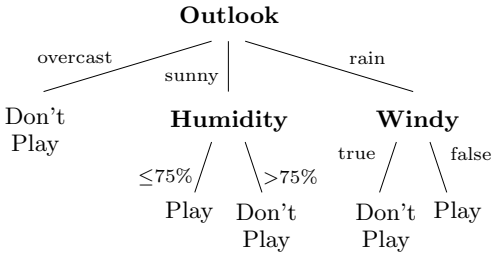


Figure 3.2: *Decision tree induced from the golf example data set.*

The introduced classification models in this section, employ an entropy measure for the derivation of its classification trees. With respect to nonlinear feature selection, we want go one step further and utilize the generalized mutual information, which will be defined in Chapter 5.

3.7 Backward Elimination- and Forward Selection Techniques

In the previous sections, an overview of various state-of-the-art techniques of feature extraction and -selection has been presented. In this context, feature extraction is always associated with a transformation of the available measurement variables.
Feature selection, in contrast, preserves the original data sequences and delivers a mere subset of the actual relevant features. This is in particular interesting for questions where data collection is extremely cost intensive, e.g. in clinical practise. Against the background of cost efficiency in health-care systems, it would be unjustifiable from the payer's perspective to collect large data sets of partially redundant clinical parameters. Besides that, patients would be unnecessarily stressed due to additional physical examinations.

Since the focus of this work is on the analysis of high dimensional, multivariate data, a feasible framework with a wide rage of applicability has to

be developed. In this context, the feature selection techniques Backward Elimination and Forward Selection, will be combined with the so called general mutual information. In this section, the two selection algorithms will be formally introduced. Against the background of a flexible feature selection framework for high dimensional, multivariate data, its advantages and drawbacks will be anticipated.

In order to simplify matters at the moment, let us define both selection methods with an abstract function $I(x, y)$. This function measures the degree of importance or information content of variable x for the determination of variable y. Since this function will guide the feature selection process, both variables are supposed to be multivariate. In Chapter 5, a sophisticated nonlinear measure of statistical dependence will be introduced on the basis of information theory.

Backward Feature Elimination

Let us suppose we have to perform feature selection on a set of N input variables. The backward elimination algorithm starts out with a complete set of input variables and successively removes specific variables. A mathematical outline of the iterative backward elimination strategy for $d = N, ..., 2$ is given in the following:

$$\begin{aligned} x_{back}(d) &= \{x_1, ..., x_d\}\,, \\ x_{back}(d-1) &= x_{back}(d) \setminus \arg\min_{v_k} \left[I(x_{back}(d), y_1) - I(v_k, y_1)\right], \end{aligned} \tag{3.39}$$

where v_k is defined as $v_k = \{x_i \mid x_i \in x_{back}(d) \backslash \{x_{i_k}\},\ x_{i_k} \in x_{back}(d)\}$ and $k = 1, ..., d$.

The removal of variables is done due to the policy described in Equation 3.39. In each iteration, every variable in $x_{back}(d)$ is temporarily removed and the information content $I(v_k, y_1)$ of the remaining variables is calculated. After this, the variable which implies the least information loss with respect to $I(x_{back}(d), y_1)$ is chosen to be removed for good, yielding $x_{back}(d-1)$. Finally, the iteration process stops if the set of input variables contains just one single variable.
The order, in which the variables are removed during the iteration, implies their relative importance. Since the variables are removed in such a way

that a minimum loss of information is generated from iteration to iteration, the most unimportant variables are removed right at the beginning. In other words, the variables providing the smallest information gain are removed first, while those with the highest information gain are removed at last. The consequence of this approach is that the elimination process has to be completed before the most relevant variables can be identified. A detailed outline of the backward elimination algorithm in pseudo-code is given in Appendix B.2.

The major drawback of the backward elimination strategy is, that it has to start out with the maximum dimension and run through all iterations before the final result can be obtained. In particular, this could cause considerable restrictions when analyzing high dimensional input data, such as demonstrated in Section. 7.2.3. In order to avoid such imponderability, the forward selection strategy has been considered as a feasible alternative approach.

Forward Feature Selection

In contrast to backward elimination, the forward selection strategy starts out with an empty set and successively adds the selected features. The forward selection strategy is in this context a fairly straightforward approach, because it employs the idea of maximizing the information gain as an essential criterion for feature selection.

Suppose that we have to identify d dimensional feature subsets from an N dimensional superset of available input variables. A mathematical outline of the iterative forward selection strategy for $d = 1, ..., N$ is given in the lines below

$$\begin{aligned} x_{forw}(0) &= \{\,\}\,, \\ x_{forw}(d) &= x_{forw}(d-1) \;\cup\; \arg\max_{v_k} \left[I(v_k, y_1) \right], \end{aligned} \tag{3.40}$$

where v_k is defined as $v_k = \{x_i \mid x_i \in x_{forw}(d-1) \cup \{x_k\}\}$. The subindex $k = 1, ..., N$ indicates the temporarily added variable.

The forward selection strategy commences with the empty feature set $x_{forw}(0)$ to which the selected variables are added. In each iteration d, every available variable x_k is temporarily added to the current set of features and the related degree of information $I(v_k, y_1)$ is calculated. The variable with the highest information value also generates the utmost information

gain with respect to the preceding iteration $(d-1)$. This particular variable is then concluded to be a valuable feature and is therefore added to the feature set $x_{forw}(d)$. An explicit outline of the forward selection algorithm is presented in Appendix B.3.

Since forward selection starts out with an empty set, this strategy does not necessarily need to run through the complete iteration process to produce a final result. The termination criterion can thus be formulated very flexible with respect to the underlying problem. There are several possible alternatives to formulate a termination criterion for the forward selection strategy.

Depending on the way of looking at a specific problem, the forward selection procedure might be carried out until either

- the maximum information gain in each iteration drops below some predefined value or
- a previously defined threshold for the total information content is reached or
- a predefined threshold for the number of selected features is reached.

In this chapter, various data analysis techniques for feature extraction and -selection have been presented to provide a detailed overview over this field. In the course of this work, it will be shown how the mentioned general mutual information can be employed to implement an effective feature selection procedure. The basic principle of this sophisticated and powerful method are described in detail in Chapter 5. Its application will be demonstrated and thoroughly discussed in Chapters 6 and 7.

4 Identification of Technical Processes with Neural Networks

This chapter provides definitions of terms concerning neural network components, various network topologies and appropriate training algorithms. Neural networks will be used as a modelling framework for system identification and the Proof-of-Concept of the outcomes of the GMI-method in Chapter, and will thus be introduced in this section.
In the first section the basic elements of neural networks are presented. The successive section provides a brief survey of commonly used network topologies. Section 4.2 is dedicated to the presentation of various training algorithms for neural structures. For a profound introduction to the subject of neural networks, the interested reader is referred to the publications of BISHOP [MB99] or ZELL [Zel94].

4.1 Network Components

Inspired by the incredible complex structures of biological systems, neural networks are massively parallel information processing systems that rely on dense arrangements of interconnections and surprisingly simple processors. In the following, all elements of neural networks are explained in detail.

Neurons

Neurons are the basic computing elements of neural networks. Figure 4.1 depicts an extended structure of the mathematical neuron model proposed by McCULLOCH and PITTS [MP43]. It receives the output of other neurons as input via weighted connections $w_{i,j}^{x}$, with $x \in \{f, r\}$. If a weight is

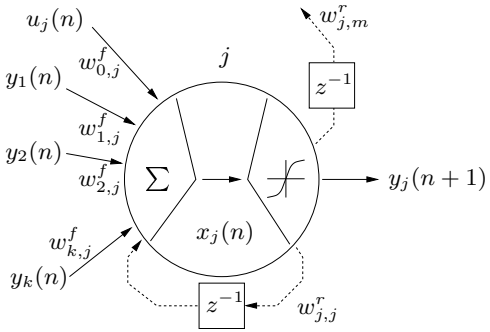


Figure 4.1: *Internal structure of a computing neuron.*

delaying the information propagated along its path, the weight is referred to as a recurrent weight connection. In the case of static networks, no recurrent weights are present.

A computing neuron includes a propagation function, an internal state and a nonlinear activation function. The inputs from other neurons together with the external input $u_j(n)$ to neuron j are mapped to the internal state $x_j(n)$ with a so called propagation function. The activation function $\sigma(x_j(n))$ relates the internal state $x_j(n)$ to the neuron output $y_j(n+1)$ in the next time step.

Weights

A directional, weighted edge $w^x_{i,j}$ from neuron i to neuron j is denoted a neural weight connection or simply weight in the terminology of neural networks. It can be interpreted as the cohesion between two neurons and represents a very crude mathematical analogy to the synaptic strengths in biological structures.
The artificial neural networks in this work incorporate two types of weights, feedforward and recurrent weights. The recurrent weights $w^r_{j,j}$ and $w^r_{j,m}$ in Fig. 4.1 are associated with the block z^{-1} indicating a time delay. Any information passing through a recurrent weight connection is delayed for a certain amount of time before reentering the same or different neurons

of the network. Since the states $x_j(n)$ of the neural structure evolve in time, the response of the network is now also depending on the network states from previous time steps. Hence, introducing recurrence in general is the major key for the ability to model dynamic behavior with neural networks. For the purpose of classification or nonlinear function approximation, no dynamic behavior has to be modelled and thus no recurrent weight connections are required.

Propagation Function

The propagation function maps the inputs to a neuron j to its internal state $x_j(n)$. This is commonly done by computing the weighted sum of all inputs in the current time step:

$$x_j(n) = \sum_{i=1}^{k} (w_{i,j}^{x} \; y_i(n)) + u_j(n), \tag{4.1}$$

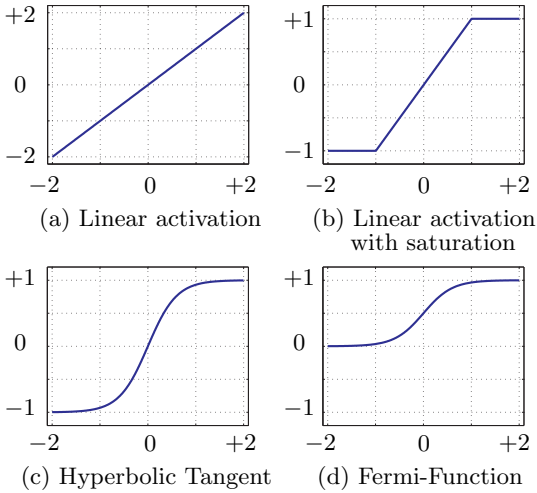
where:

$y_i(n)$: output of neuron i in time step n
$u_j(n)$: external input to neuron j in time step n
$w_{i,j}^{x}$: weight connection from neuron i to neuron j.

It should be mentioned that more propagation functions than the weighted sum of inputs do exist in literature. They have been thoroughly investigated by the neural community over the past decades. One particular interesting neural network uses so called $(\Sigma - \Pi)$ neuron model ([NS96]). Since this work is only concerned with the introduced neuron model according to Fig. 4.1, the interested reader is referred to the specific literature.

Activation Function

The activation- or squashing function $\sigma(\cdot)$ maps the internal state x_j of a neuron to its according output y_j. Figure 4.2 depicts the four most commonly used activation functions. For the neuron model introduced above, any non constant, monotone increasing, bounded and continuously differentiable nonlinear function $\sigma : I\!R \mapsto I\!R$ is a reasonable choice.
It has to be mentioned that for other networks types, eg. Radial Basis Function (RBF) Networks, the characteristics of the mapping function might be different from those stated in this section.



(a) Linear activation (b) Linear activation with saturation

(c) Hyperbolic Tangent (d) Fermi-Function

Figure 4.2: *Activation functions for artificial neurons.*

The linear activation function in Fig. 4.2(a),

$$\sigma_{lin}(x) = ax + b, \quad a, b \in \mathbb{R}, \tag{4.2}$$

is extensively used for input neurons in order to scale an external input to a standard interval. This prevents involuntary weighting of the input signal according to its absolute value.

Figure 4.2(b) shows a linear activation function with saturation,

$$\sigma_{sat}(x) = \begin{cases} -b & : \quad x < -a \\ \frac{b}{a}x & : \quad -a \leq x \leq a, \\ b & : \quad x > a, \end{cases} \qquad a, b \in \mathbb{R}_0^+ \tag{4.3}$$

This function is not continuously differentiable within its entire domain. Since training algorithms like Back-Propagation or Real Time Recurrent Learning (see [Zel94]) are gradient descent methods and require the derivative of the employed activation functions to be computed, this choice yields quite cumbersome formulations in learning algorithms. The advantage of

this type of function is the relatively high sensitivity around zero and its bounded range.

The most commonly used activation functions for computing neurons, i.e. neurons in hidden- or output layers, are depicted in Fig.4.2(c) and Fig.4.2(d). The hyperbolic tangent in Fig.4.2(c),

$$\sigma_{tanh}(x) = \frac{a}{2} \tanh\left(\frac{x}{\theta}\right) + b, \quad a, b, \theta \in I\!R, \tag{4.4}$$

is differentiable in its entire domain and bounded in range. Differentiation with respect to the function variable $x \in I\!R$ yields

$$\sigma'_{tanh}(x) = \frac{2a}{\theta \left(e^{\frac{x}{\theta}} + e^{-\frac{x}{\theta}}\right)^2}, \quad a, \theta \in I\!R. \tag{4.5}$$

Figure 4.2(d) depicts the Fermi-function

$$\sigma_{fermi}(x) = \frac{a}{\left(1 + e^{-\frac{x}{\theta}}\right)} + b, \quad a, b, \theta \in I\!R. \tag{4.6}$$

Like in the case of the hyperbolic tangent, the Fermi-Function is continuously differentiable and its derivative is easily determined to be

$$\sigma'_{fermi}(x) = \frac{a}{\theta} \left(\frac{1}{1 + e^{-\frac{x}{\theta}}}\right) \left(1 - \frac{1}{1 + e^{-\frac{x}{\theta}}}\right), \quad a, \theta \in I\!R. \tag{4.7}$$

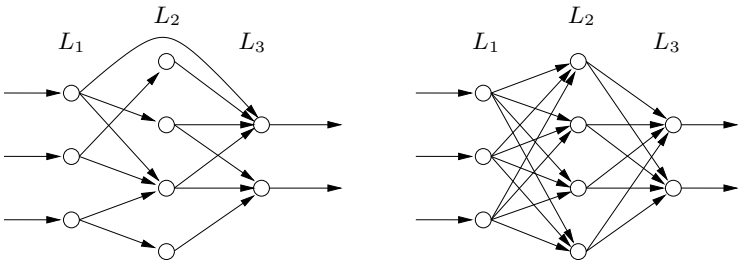
To meet the requirements of special training problems, the range and the translational displacement of both activation functions can be adjusted with the parameters a and b. The parameter θ controls the shape of the employed squashing functions. Usually the range of the hyperbolic tangent and the Fermi-Function is restricted to the interval $[-1, 1]$ and $[0, 1]$, respectively.

4.2 Network Topology

Based upon the definitions in Sec.4.1, neural structures are constructed by connecting the basic computing elements with forward or recurrent weights. In general, a neural network can be regarded as a directed, weighted and possibly cyclic graph. The neurons are interpreted as nodes and the weight connections $w^x_{i,j}$ correspond to the directed edges of the graph. Over the past years of research in neural computing, numerous network topologies have been developed and investigated. Basically all existing topologies can be classified into two categories — feedforward- and recurrent neural structures. In the following, some representative examples of the network topologies which are employed in this work will be introduced.

4.2.1 Feedforward Structures

A feedforward neural network has no cycles or loops in its graph representation. The neurons are grouped into layers and only non delaying weight connections exist between neurons of successive layers.



(a) Simple Feedforward Network (b) Multilayer Perceptron

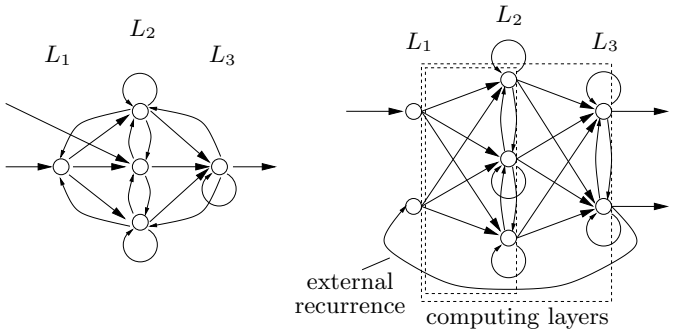
Figure 4.3: *Graph Representations of Feedforward Neural Networks.*

Figure 4.3(a) depicts an exemplary representation of a simple feedforward neural network. The information is propagated from the input layer L_1 through the hidden layer L_2 to the output layer L_3. In Fig. 4.3(b), a Multilayer Perceptron (MLP) with one hidden layer is shown. The neurons of successive layers are completely interconnected with feedfoward weights. It can be proved that MLPs with at least one hidden layer are able to approximate any continuously differentiable and Borel-measurable function with

theoretically unlimited accuracy [HSW89]. In practice however, the quality of approximation depends on various factors, e.g. the number of computing neurons, a proper training set with reasonably chosen input vectors and the employed learning algorithm. Since feedforward neural structures do not have recurrent weight connections among internal neurons, the response of the network in the next time step does not depend on the current state of the neurons. Hence this class of network is not capable of identifying dynamic processes without the additionally introducing the aspect of time, eg. through tapped delay lines or the application of specific learning algorithms.

4.2.2 Recurrent Structures

Dynamic neural structures differ from conventional feedforward networks in the way of how previous output is reused as a part of the input signal in successive time steps. Due to the recurrent weight connections, the internal state and thus the output of each neuron shows a time depended behavior. When confronted with the same input in different time steps, each particular neuron and hence the entire neural network will respond with different outputs.



(a) Simple recurrent network (b) Recurrent Multilayer Perceptron

Figure 4.4: *Graph representations of dynamic neural structures.*

In Fig. 4.4, two types of recurrent neural structures are depicted. Figure 4.4(a) depicts an example of a simple recurrent network with no layered structure and thus no specific information propagation order. A special network topology, known as Recurrent Multilayer Perceptron (RMLP), is shown in Fig. 4.4(b). Similar to feedforward networks, this type shows a layered structure which defines the flow of information through the network. Recurrent Multilayer Perceptrons have been proposed by PUSKORIUS and FELDKAMP [PF94] for the identification and control of dynamic, nonlinear systems.
A RMLP consists of three different types of connections: forward, recurrent and external recurrent weights. The nodes of consecutive layers are connected with forward weights, while neurons belonging to one particular layer are fully interconnected with recurrent weights. In addition, the response of the global network can be fed back into the input layer along external recurrent weight connections.
Due to the introduction of recurrent connections, this class of neural structures is capable of learning time dependent behavior without further modification. Since recurrent networks are oscillatory structures, this might lead to unstable behavior during training and application. Along with the growing complexity, the employed training algorithms become very expensive with respect to computational costs. In the following section, various algorithms are presented for the training of feedforward- and recurrent neural networks.

4.3 Neural Training Algorithms

This section is dedicated to the presentation of some commonly used algorithms for supervised gradient descent training based on the neural model introduced above. Depending on the class and internal structure of the underlying network, different algorithms have to be applied. The intention of this section is to show the rapidly increasing complexity of the learning algorithms when it comes to recurrent neural structures. In a later chapter, a new learning algorithm will be proposed which will work for both – feedforward and recurrent neural networks.

4.3.1 Back Propagation

Back Propagation is the classical gradient based training algorithm for neural networks. The conceptual basis of Back Propagation was first presented in 1960 by ROSENBLATT [Ros60], then reinvented by David Parker

in 1982, and finally presented to a wide readership in 1986 by RUMMELHART and McCLELLAND [RM86]. Early applications of of Back Propagation were done by Sejnowski and Rosenberg at Johns Hopkins University. Typically, Back Propagation is applied for MLP networks with one or more hidden layers.

The output of a particular neuron in a MLP is governed by the following equation:

$$y_j = \sigma(x_j) = \sigma\left(\sum_i w_{i,j}\, y_i\right). \tag{4.8}$$

In Equation 4.8, the output y_j of neuron j is the weighted sum of all inputs to this neuron, passed through the activation function σ.
All gradient descent algorithms are based on the minimization of a cost- or error function

$$E = \frac{1}{2}\sum_j (t_j - y_j)^2\,, \tag{4.9}$$

where t_j is the target- and y_j is the actual output value of the j-th neuron in the output layer. The output and thus the error of an output neuron depends on the weights of the neural network. The adaption of one particular weight $w_{i,j}$ is proportional to the negative gradient of the error function and a factor η depicting the learning rate.

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} \tag{4.10}$$

Application of the chain rule to the error function E yields

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial x_j}\,\frac{\partial x_j}{\partial w_{i,j}}. \tag{4.11}$$

The second factor in Equation 4.11 is determined by executing the differentiation of x_j with respect to a particular weight $w_{i,j}$:

$$\frac{\partial x_j}{\partial w_{i,j}} = \frac{\partial}{\partial w_{i,j}}\left(\sum_i w_{i,j}\, y_i\right) = y_i. \tag{4.12}$$

The first factor of Equation 4.11 is defined as the error value δ_j of neuron j. During training, this value will be propagated back through the neural network.

$$\delta_j = -\frac{\partial E}{\partial x_j}\,\frac{\partial x_j}{\partial y_j} \tag{4.13}$$

The second partial derivative in Equation 4.13 is determined by differentiating Equation 4.8 with respect to x_j:

$$\frac{\partial o_j}{\partial x_j} = \frac{\partial(\sigma(x_j))}{\partial x_j} = \sigma(x_j)'. \tag{4.14}$$

The only term left to be determined is now the first factor of Equation 4.13. In order to computed the sensitivity of the error function E with respect to the actual output y_j of a particular neuron, two cases have to be distinguished.

- Neuron j is part of the output-layer:

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\frac{1}{2}\sum_k (t_k - y_k)^2\right) = (t_j - y_j). \tag{4.15}$$

- Neuron j is part of a hidden layer. In this case the required partial derivative can only be determined indirectly:

$$\begin{aligned}\frac{\partial E}{\partial y_j} &= -\sum_k \frac{\partial E}{\partial x_k}\,\frac{\partial x_k}{\partial y_j} \\ &= \sum_k \delta_k \frac{\partial}{\partial y_j}\sum_i w_{i,k}\, y_i = \sum_k \delta_k\, w_{j,k}\end{aligned} \tag{4.16}$$

Finally, utilizing the results of Equations 4.11, 4.12 and 4.13 in Equation 4.10, the online update rule for a particular weight $w_{i,j}$ can be formulated as

$$\Delta w_{i,j} = \eta\, y_i\, \delta_j, \tag{4.17}$$

where

$$\delta_j = \begin{cases} \sigma'(x_j)\,(t_j - y_j), & \text{Neuron j is part of the output layer} \\ \sigma'(x_j)\sum_k \delta_k\, w_{j,k}, & \text{Neuron j is part of a hidden layer.} \end{cases}$$

After each propagation of an input pattern, the output error of every neuron is determined as the difference between the target- and the output pattern. Figure 4.5 depicts the back propagation of the error value δ_j while updating the weight connections according to Equation 4.17.

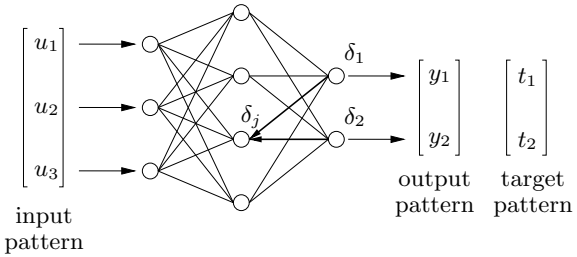


Figure 4.5: *The standard back propagation algorithm.*

4.3.2 Back Propagation Through Time

Back Propagation in its basic version can only be applied to feedforward networks. If the employed networks show recurrent weight connections, a modification called Back Propagation Through Time (BPTP) has to be used.

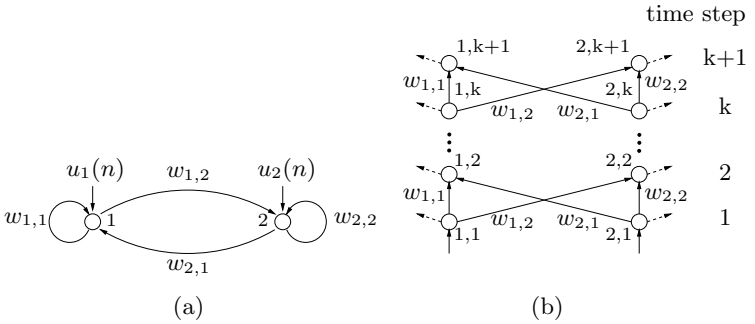


Figure 4.6: *Recurrent neural network and its feedforward equivalent.*

Figure 4.6(a) depicts a small recurrent network with two computing nodes. The neuron dynamics is governed by the following equation:

$$y_j(n+1) = \sigma(x_j(n+1)) = \sigma\left(\sum_i \left(w_{i,j} y_i(n) + u_j(n+1)\right)\right). \qquad (4.18)$$

MINSKY and PAPERT [MP88] showed that every recurrent network can be transformed into an equivalent feedforward network by a technique known as *unfolding in time*. In order to perform this transformation, the length of the training sequence has to be known a priori. In every instant of time, neurons and weights of the original recurrent structure are substituted with additional components, yielding the feedforward network in Fig. 4.6(b). The neurons in layer k are now reflecting the neuron states of the original network in time step k. Since identical weights in Fig. 4.6(b) correspond to a particular weight connection in Fig. 4.6(a), they have to be set to identical values. In order to perform training on the transformed network, the weight changes are accumulated and applied after the presentation of the entire training set. For a detailed outline of the BPTT algorithm refer to ZELL [Zel94].
The major drawbacks of this algorithm are the restriction to a fixed number of time steps and the considerable amount of memory for the transformed network. Assuming a time horizon of length m and a network comprised of n neurons, the memory requirement is in the complexity class $O(mn^2)$. Since a general framework is required for dynamic neural structures, training algorithms with true online capability have to be considered.

4.3.3 Real Time Recurrent Learning

Real Time Recurrent Learning (RTRL) is, as well as Back Propagation, a gradient based learning algorithm for layered neural networks. It is a variant of Back Propagation for recurrent networks, which has been proposed by WILLIAMS and ZIPSER [WZ89].

The output of a particular neuron is defined in Equation 4.18. Similar to Equation 4.10 this yields, in conjunction with Equation 4.9, for the adaption of a particular weight $w_{i,j}$:

$$\begin{aligned}
\Delta w_{i,j}(n) &= -\eta \frac{\partial E(n)}{\partial w_{i,j}} = -\eta \frac{\partial}{\partial w_{i,j}} \left(\frac{1}{2} \sum_k \left(E_k(n) \right)^2 \right) \\
&= -\frac{1}{2} \eta \sum_k \frac{\partial \left(\left(t_k(n) - y_k(n) \right)^2 \right)}{\partial w_{i,j}} \\
&= \eta \sum_k E_k(n) \frac{\partial y_k(n)}{\partial w_{i,j}}, \qquad (4.19)
\end{aligned}$$

where k is referencing the output neurons of the network.

Depending on the structure of the underlying network, two methods for determining the partial derivatives in Equation 4.19 are presented. If layered recurrent networks with arbitrarily connected neurons are employed, the required derivatives can be obtained directly by differentiation of Equation 4.18 with respect to a particular weight $w_{i,j}$.

$$\begin{aligned}
\frac{\partial y_k(n)}{\partial w_{i,j}} &= \frac{\partial}{\partial w_{i,j}} \left(\sigma \left(\sum_m w_{m,k} y_m(n-1) + u_k(n) \right) \right) \\
&= \sigma'(x_k(n)) \left[\frac{\partial}{\partial w_{i,j}} \left(\sum_m w_{m,k} y_m(n-1) \right) + \frac{\partial(u_k(n))}{\partial w_{i,j}} \right] \\
&= \sigma'(x_k(n)) \left[\sum_m \frac{\partial(w_{m,k})}{\partial w_{i,j}} y_m(n-1) + \sum_m w_{m,k} \frac{\partial(y_m(n-1))}{\partial w_{i,j}} \right] \\
&= \sigma'(x_k(n)) \left[\delta^*_{k,j}\, y_i(n-1) + \sum_m w_{m,k} \frac{\partial(y_m(n-1))}{\partial w_{i,j}} \right] \qquad (4.20)
\end{aligned}$$

In Equation4.20, $\delta_{k,j}$ denotes the Kronecker symbol:

$$\delta^*_{k,j} =: \begin{cases} 1, & k = j \\ 0, & else. \end{cases} \qquad (4.21)$$

When employing RMLPs, a specific but rather complex method for obtaining dynamic derivatives is used. PUSKORIUS and FELDKAMP [PF94] presented an effective approach for the computation of the required partial derivatives. Since the RMLP in Fig. 4.4(b) can be seen as a concatenation of consecutive subnetworks, the spatial dependencies between particular neurons can be taken into account for the derivation of a learning algorithm.
The output $y_{i,j}(n)$ of a particular neuron j in layer i is a function of the output vector $\vec{y}_{i-1}(n)$ of the preceding layer in time step n, the output vector $\vec{y}_i(n-1)$ of the actual layer in time step $n-1$ and the weight vector $\vec{w}_i$ of subnetwork i.

The neuron dynamics in a RMLP network are now given by the following equation:

$$y_{i,j}(n) = F\left(\vec{y}_{i-1}(n), \vec{y}_i(n-1), \vec{w}_i\right), \qquad (4.22)$$

where

$$F\left(\vec{y}_{i-1}(n), \vec{y}_i(n-1), \vec{w}_i\right) = \sigma\left(\sum_{p=1}^{N_{i-1}} w_{p,j}^{f,i} y_{i1,p}(n) + \sum_{p=1}^{N_i} w_{p,j}^{r,i} y_{i,p}(n-1)\right) \tag{4.23}$$

and

$\vec{y}_{i-1}(n)$	is the output vector of subnet $i-1$ at time step n,
$\vec{y}_i(n-1)$	the output vector of subnet i at time step $n-1$,
$\vec{w}_i$	the weight vector of subnet i,
$y_{i,j}$	the output of neuron j in layer i,
$w_{k,j}^{r,i}$	the recurrent weight from neuron k to neuron j in layer i and
$w_{k,j}^{f,i}$	the feedforward weight from neuron k in layer $i-1$ to neuron j in layer i.

Similar to Equation 4.19, the rule for the adaption of a particular weight $w_{k,j}^{x,g}$ can be formulated as:

$$\Delta w_{k,j}^{x,g} = -\eta \frac{\partial E(n)}{\partial w_{k,j}^{x,g}} = \eta \sum_{p=1}^{N_l} E_p(n) \frac{\partial y_{l,p}(n)}{\partial w_{k,j}^{x,g}}, \tag{4.24}$$

where N_l is the number of neurons in the output layer of the RMLP and

$$x = \begin{cases} r, & \text{for recurrent weights} \\ f, & \text{for forward weights.} \end{cases}$$

Since the dynamics of the employed neurons take into account the spatial dependencies among the elements of a RMLP, the determination of the required partial derivatives is rather complex. Differentiation of Equation 4.23 with respect to a particular weight $w_{k,j}^{x,g}$ and application of the general chain rule yields:

$$\begin{aligned} \frac{\partial y_{i,j}(n)}{\partial w_{k,j}^{x,g}} &= \frac{\partial\left(F(\vec{y}_{i-1}(n), \vec{y}_i(n-1), \vec{w}_i)\right)}{\partial w_{k,j}^{x,g}} \\ &= \frac{\partial\left(F(\vec{y}_{i-1}(n), \vec{y}_i(n-1), \vec{w}_i)\right)}{\partial \vec{y}_{i-1}(n)} \frac{\partial \vec{y}_{i-1}(n)}{\partial w_{k,j}^{x,g}} + \\ &\quad + \frac{\partial\left(F(\vec{y}_{i-1}(n), \vec{y}_i(n-1), \vec{w}_i)\right)}{\partial \vec{y}_i(n-1)} \frac{\partial \vec{y}_i(n-1)}{\partial w_{k,j}^{x,g}} + \end{aligned}$$

$$
\begin{aligned}
&+\frac{\partial\left(F(\vec{y}_{i-1}(n),\vec{y}_i(n-1),\vec{w}_i)\right)}{\partial\vec{w}_i}\frac{\partial\vec{w}_i}{\partial w_{k,j}^{x,g}}\\
&=\frac{\partial y_{i,j}(n)}{\partial\vec{y}_{i-1}(n)}\frac{\partial\vec{y}_{i-1}(n)}{\partial w_{k,j}^{x,g}}+\frac{\partial y_{i,j}(n)}{\partial\vec{y}_i(n-1)}\frac{\partial\vec{y}_i(n-1)}{\partial w_{k,j}^{x,g}}+\frac{\partial y_{i,j}(n)}{\partial\vec{w}_i}\frac{\partial\vec{w}_i}{\partial w_{k,j}^{x,g}}\\
&=\sum_{p=1}^{N_{i-1}}\frac{\partial y_{i,j}(n)}{\partial y_{i-1,p}(n)}\frac{\partial y_{i-1,p}(n)}{\partial w_{k,j}^{x,g}}+\sum_{p=1}^{N_i}\frac{\partial y_{i,j}(n)}{\partial y_{i,p}(n-1)}\frac{\partial y_{i,p}(n-1)}{\partial w_{k,j}^{x,g}}+\\
&+\delta_{g,i}\,\frac{\partial y_{i,j}(n)}{\partial w_{k,j}^{x,g}}. \qquad (4.25)
\end{aligned}
$$

In the last term of Equation4.25, the symbol $\delta_{g,i}$ denotes the Kronecker symbol. The partial derivatives $\frac{\partial y_{i,j}(n)}{\partial y_{i-1,p}(n)}$, $\frac{\partial y_{i,j}(n)}{\partial y_{i,p}(n-1)}$ and $\delta_{g,i}\frac{\partial y_{i,j}(n)}{\partial w_{k,j}^{x,g}}$ in Equation 4.25 are found to be:

$$
\begin{aligned}
\frac{\partial y_{i,j}(n)}{\partial y_{i-1,p}(n)} &= \frac{\partial(\sigma(x_{i,j}(n)))}{\partial y_{i-1,p}(n)}\\
&=\sigma'(x_{i,j}(n))\frac{\partial}{\partial y_{i-1,p}(n)}\left(\sum_{t=1}^{N_{i-1}}y_{i-1,t}(n)w_{t,j}^{f,i}+\sum_{t=1}^{N_i}y_{i,t}(n)w_{t,j}^{r,i}\right)\\
&=\sigma'(x_{i,j}(n))w_{p,j}^{f,i}, \qquad (4.26)\\
\frac{\partial y_{i,j}(n)}{\partial y_{i,p}(n-1)} &= \sigma'(x_{i,j}(n))w_{p,j}^{r,i} \quad \textit{and} \qquad (4.27)\\
\delta_{g,i}\frac{\partial y_{i,j}(n)}{\partial w_{k,j}^{x,g}} &= \begin{cases} 0 & \text{, if } g\neq i\\ \frac{\partial}{\partial w_{k,j}^{r,g}}\left(\sigma(x_{i,j}(n))\right) & \text{, if } g=i\end{cases}\\
&= \begin{cases} 0 & \text{, if } g\neq i\\ \sigma'(x_{i,j}(n))\frac{\partial}{\partial w_{k,j}^{r,g}}\left(net_{i,j}(n)\right)) & \text{, if } g=i\end{cases}\\
&= \begin{cases} 0 & \text{, if } g\neq i\\ \sigma'(x_{i,j}(n))y_{i,k}(n) & \text{, if } g=i.\end{cases} \qquad (4.28)
\end{aligned}
$$

Since the employed training sequences can be of arbitrary length, the presented version of RTRL depicts a true online learning algorithm for layered recurrent neural networks.

4.3.4 Extended Kalman Filter Training

Apart from pure gradient descent algorithms, the training of neural networks can also be performed by using methods from estimation theory. The basic Kalman Filter (KF) is designed for parameter- or state estimations in linear, dynamic systems. If the system turns out to be nonlinear, a linearization around the current point of estimate has to be performed, yielding the Extended Kalman Filter (EKF). For a detailed derivation and outline of the EKF refer to CHUI [CC87] or LOFFELD [Lof90]. In the following, the EKF and its applicability to neural networks will be presented.

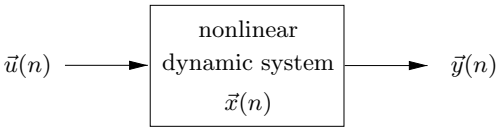


Figure 4.7: *General description of a nonlinear dynamic system model.*

Figure 4.7 generally depicts the model of a time discrete nonlinear dynamic system of the form:

$$\vec{x}(n+1) = f[\vec{x}(n), \vec{u}(n)] + \vec{\xi}(n), \tag{4.29}$$

$$\vec{y}(n) = g[\vec{x}(n), \vec{u}(n)] + \vec{\eta}(n). \tag{4.30}$$

Here the vector $\vec{x}(n)$ is the state of the dynamic system at time n, $\vec{u}(n)$ is the input and $\vec{y}(n)$ the observation at the system output. The system and the observation noise $\vec{\xi}(n)$ and $\vec{\eta}(n)$ are assumed to be zero mean Gaussian white noise sequences. The nonlinear vector functions $f(\cdot)$ and $g(\cdot)$ depict the state transition- and the observation function of the dynamic system. The EKF is commonly used as a stochastic observer in physical system models, e.g. in radar tracking systems, for the optimal estimation of not directly measurable system variables. With this approach, a neural network during training is regarded as a dynamic system and its weights as the parameters to be estimated. For this constellation, SINGHAL and WU [SW89] showed the applicability of the Extended Kalman Filter as a training algorithm for neural structures. Compared to conventional gradient based training algorithms, the EKF successively estimates the weights

based on the deviation $[\vec{d}(n) - h(\vec{\hat{w}}(n), \vec{u}(n))]$ of the neural structure from the desired output $\vec{d}(n)$ [HM99]. This statistical method minimizes the expectation value of the squared error between the estimated weight vector $\vec{\hat{w}}$ and the weight vector solving the training problem. The EKF is proved for linear systems to provide an optimal estimate for the weight vector which satisfies

$$\lim_{n\to\infty} E\left[\left(\vec{w}(n) - \vec{\hat{w}}(n)\right)^T \cdot S \cdot \left(\vec{w}(n) - \vec{\hat{w}}(n)\right)\right] = 0. \tag{4.31}$$

This states that the expectation value of the squared error between the required weight vector and its estimate converges to zero. For nonlinear systems like neural networks, linearization around the current point of estimate has to be performed. Due to this fact, one can only expect to find a weight estimate $\vec{\hat{w}}$, which yields a local minimum of Equation 4.31.

The EKF equations utilized for the weight estimation of neural networks are:

$$\begin{aligned} P(n+1) &= P(n) - K(n) \cdot H^T(n) \cdot P(n) + Q(n) \\ K(n) &= P(n) \cdot H(n) \cdot \left[(\eta(n) \cdot S(n))^{-1} + H^T(n) \cdot P(n) \cdot H(n)\right]^{-1} \\ \vec{\hat{w}}(n+1) &= \vec{\hat{w}}(n) + K(n) \cdot \left(\vec{d}(n) - h(\vec{\hat{w}}(n), \vec{u}(n))\right), \quad \text{with} \end{aligned} \tag{4.32}$$

$$\vec{d}(r) = \begin{pmatrix} d_1(r) \\ d_2(r) \\ \vdots \\ d_m(r) \end{pmatrix}$$

is the desired output vector of the neural network in time step r.

$$h(\vec{\hat{w}}(r), \vec{u}(r)) = \begin{pmatrix} h_1(\vec{\hat{w}}(r), \vec{u}(r)) \\ h_2(\vec{\hat{w}}(r), \vec{u}(r)) \\ \vdots \\ h_m(\vec{\hat{w}}(r), \vec{u}(r)) \end{pmatrix}$$

describes the output vector of the neural network, depending on the estimation of the weight vector $\vec{\hat{w}}(r)$ in time step r. $h(\cdot)$ represents the response of the neural network.

$$\vec{\hat{w}}(r) = \begin{pmatrix} \hat{w}_1(r) \\ \hat{w}_2(r) \\ \vdots \\ \hat{w}_n(r) \end{pmatrix}$$

is the estimation of the weight vector in time step r, comprising all weights of the neural network.

$$H(r) = \begin{pmatrix} \frac{\partial y_1(r)}{\partial w_1(r)} & \cdots & \frac{\partial y_m(r)}{\partial w_1(r)} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1(r)}{\partial w_n(r)} & \cdots & \frac{\partial y_m(r)}{\partial w_n(r)} \end{pmatrix}$$

depicts the Jacobian matrix, which serves as a linear approximation of the neural network around the current point of estimate.

$$K(r) = \begin{pmatrix} k_{1,1}(r) & ... & k_{1,m}(r) \\ \vdots & \ddots & \vdots \\ k_{n,1}(r) & ... & k_{n,m}(r) \end{pmatrix}$$

is the Kalman-Gain matrix, used for updating the covariance matrix $P(r)$.

$$P(r) = \begin{pmatrix} p_{1,1}(r) & ... & p_{1,n}(r) \\ \vdots & \ddots & \vdots \\ p_{n,1}(r) & ... & p_{n,n}(r) \end{pmatrix}$$

represents the error covariance matrix. The elements of $P(r)$ describe the covariance among the weights, based on all previous estimation errors.

$$Q(r) = \begin{pmatrix} q_{1,1}(r) & ... & 0 \\ \vdots & \ddots & \vdots \\ 0 & ... & q_{n,n}(r) \end{pmatrix}$$

is the driving noise matrix, adding noise to prevent the estimation process from getting stuck in local minima. $q_{i,i} \in \left[10^{-6}, 10^{-2}\right], i = 1, ..., n$

$$S(r) = \begin{pmatrix} s_{1,1}(r) & ... & s_{1,m}(r) \\ \vdots & \ddots & \vdots \\ s_{1,m}(r) & ... & s_{m,m}(r) \end{pmatrix}$$

is a user defined, positive definite, symmetric matrix. $S(r)$ defines, in conjunction with $\eta(\cdot)$, the learning rate of the neural network.

The elements of the Jacobian matrix are obtained through Equation 4.20 or Equation 4.25. In spite of the necessity of performing a linearization, the EKF algorithm obtains considerably better solutions after fewer iterations than purely gradient based learning methods. The major drawback of the EKF is its computational complexity. The computational efforts of this algorithm include the computation of the Jacobian matrix in addition to the application of the EKF equations. Since the elements of the Jacobian matrix are identical to the gradients delivered by the RTRL algorithm, the computational expenses of this complex algorithm is extended by the EKF equations.

5 Measuring the Information Flow for Feature Selection

In Chapter 3, several feature extraction and -selection methods have been presented. This chapter is intended to introduce the concept of mutual information as the basis of an optimal feature selection framework [PFX00], [SPP06].
One of the basic postulates of information theory is that information can be treated like a measurable physical quantity, such as density or mass. This theory has been widely applied by communication engineers and some of its concepts have found application in other fields of research. Whenever entities of the real world interact, a more or less abstract flow of information occurs. Quantifying this flow of information is of vital interest for the determination of potential causalities. The concept of mutual information[1] is based on the family of Rényi-Entropies [RB61], which include Hartley's [Har28] and the well known Shannon-Entropy[2] [Sha48]. This chapter is concerned with the theoretical background of a general mutual information which is based on the second order Rényi-Entropy H_2.

5.1 Concept of Mutual Information

Information theory provides us with a sophisticated methodology for analyzing the statistical dependencies in scalar or multivariate time series. The mutual information $I(\xi, \eta)$ can be interpreted as the quantity of information that is obtainable about a random variable η, from the prior knowledge of another random variable ξ. $I(\xi, \eta)$ can be also seen figuratively as a conjunction of two information sets, where $H((\xi, \eta))$ is the entropy of the compound random variable (ξ, η). Figure 5.1 depicts the case where

[1] An alternative notation for mutual information is relative information or synentropy.
[2] Shannon used the term rate of transmission, instead of mutual information.

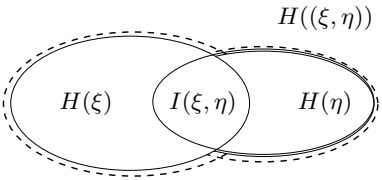


Figure 5.1: *Mutual Information* $I(\xi,\eta)$.

$I(\xi,\eta) < \min(H(\xi), H(\eta))$. In this case η could be determined only partly from ξ. On the one hand, if both random variables are statistical independent, the mutual information is equal to the empty set. On the other hand, if $I(\xi,\eta) = H(\eta)$ the mutual information $I(\xi,\eta)$ reaches its maximum. In this case, η depends completely on ξ and can hence be determined solely from the latter.

Two, possibly multivariate, signals $\{x_n\}$ and $\{y_n\}$ can be interpreted as realizations of the random variables $\{\xi_n\}$ and $\{\eta_n\}$. In this case, the mutual information is employed for measuring the flow of information and hence the degree of statistical dependency between both measurement signals. However, the mutual information does not deliver any statement about the cause or the path of information flow. For each application, this has to be determined in a separate modelling step. In our case, neural networks will be employed as a unified modelling framework.

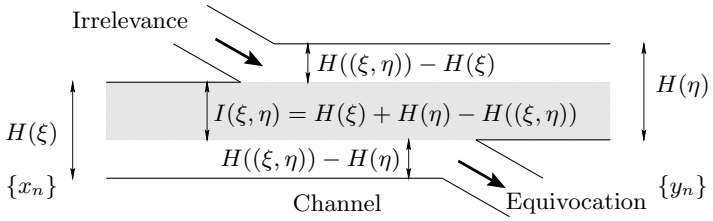


Figure 5.2: *Shannon's model for the flow of information through an abstract, symmetric message channel contaminated with noise.*

Shannon introduced the concept of mutual information for the quantitative description of an abstract message channels. Figure 5.2 depicts Shannon's model for the flow of information through an abstract, symmetric message channel contaminated with noise. The received information $H(\eta)$ is comprised of the mutual information $I(\xi, \eta)$ and the irrelevance $H((\xi, \eta)) - H(\xi)$ resulting from disturbances. $H((\xi, \eta)) - H(\eta)$ represents equivocation, i.e. the part of information which is actually lost by the channel and is never received.
In contrast to other methods of nonlinear time series analysis, based on ergodic theory like the estimation of metric entropies, fractal dimensions and Lyapunov-Exponents, the mutual information is rather universally applicable [Pom97].

For a discrete random variable ξ with probability distribution $\{p_m\}$, Shannon introduced the entropy measure H as

$$H(\xi) = -\sum_m p_m \; log \; p_m, \tag{5.1}$$

and formulated an interpretation from the viewpoint of coding theory. Later, CHINTSCHIN [Chi61] and FADDEJEW [Fad56] presented an axiomatic characterization of H.

The mutual information for two discrete random variables ξ and η is then defined as

$$I(\xi, \eta) = H(\eta) - [H((\xi, \eta)) - H(\xi)], \tag{5.2}$$

where $H(\eta)$ is a measure for the A-priori uncertainty of η and $H((\xi, \eta)) - H(\xi)$ is a measure for its remaining A-posteriori uncertainty, if ξ is known. The quantity $I(\xi, \eta)$ represents the amount of information about the random variable η, if ξ is known. Since the converse holds also, the mutual information is symmetric with respect to its arguments. An alternative formulation of Equation 5.2 is

$$I(\xi, \eta) = I(\eta, \xi) = H(\eta) - H(\eta|\xi), \tag{5.3}$$

where $H(\eta|\xi)$ is the conditional entropy of η under the condition that ξ is already known.

5.2 Theory of the General Mutual Information I_α

RÉNYI [RB61], [Rén77] introduced a more general family $\{H_\alpha(\xi)\}$ of entropy measures, where $\alpha \in \mathbb{R}$. Let ξ be a discrete random variable with the probability distribution

$$P(\xi = x_m) = \{p_m\}_{m=1}^{M}.$$

It should be mentioned, that RÉNYI considered also probability distributions with $\sum_{m=1}^{M} p_m \leq 1$. Throughout this work, only normalized and complete probability distributions are assumed: $\sum_{m=1}^{M} p_m = 1$, with $p_m \geq 0$.

The Rényi-Entropy of order $\alpha \in \mathbb{R}, \alpha \geq 0$ generalizes the Shannon-Entropy. This family of entropy measures is defined as:

$$H_\alpha(\xi) = H_\alpha(\{p_m\}_{m=1}^{M}) = \begin{cases} \frac{1}{1-\alpha} \log_2 \left(\sum_{m=1}^{M} p_m^\alpha \right) & : \quad \alpha \geq 0, \alpha \neq 1 \\ -\sum_{m=1}^{M} p_m \log_2 p_m & : \quad \alpha = 1, \end{cases} \tag{5.4}$$

with $0^0 := 0$ and $0 \log_2(0) := 0$. In order to obtain a measure for the mutual information in units of bits, the binary logarithm log_2 is commonly used[3].

If α is set to zero, Hartley's entropy measure is obtained:

$$H_0(\xi) = \log_2 \left(|\{\xi = x_m : P(x_m) \neq 0)\}| \right),$$

where $|\{\cdot\}|$ is the set cardinality of the non vanishing realizations of the random variable ξ. In case of $\alpha = 1$, the Shannon-Entropy $H_1(\xi)$ is obtained, which is continuously embedded[4] into the set of general information measures described by Equations 5.4.

[3] One bit can be regarded as the information or uncertainty, obtained from a random experiment with two outcomes of equal probability:
$H(\{\frac{1}{2}, \frac{1}{2}\}) = \frac{1}{1-\alpha} log_2 \left((\frac{1}{2^\alpha} + \frac{1}{2^\alpha}) \right) = \frac{1}{1-\alpha} \log_2 2^{1-\alpha} = 1$ [bit].

[4] L'Hospital: $\lim_{\alpha \to 1} H_\alpha(\xi) = -\sum_{m=1}^{M} p_m \log_2 p_m$

Let further be η another discrete random variable with the probability distribution

$$P(\eta = y_n) = \{q_n\}_{n=1}^{N},$$

and let

$$P((\xi = x_m, \eta = y_n)) = \{s_{m,n}\}_{m=1,n=1}^{M,N}$$

be the probability distribution of the compound random variable (ξ, η). Regarding the Rényi-Entropies, introduced in Equations 5.4 for $\alpha = 1$, the term

$$I_1(\xi, \eta) = H_1(\eta) - [H_1((\xi, \eta)) - H_1(\xi)]$$

is considered. This yields the mutual information which is based on the Shannon-Entropy:

$$I_1(\xi, \eta) = \sum_{m=1,n=1}^{M,N} s_{m,n} \log_2 \frac{s_{m,n}}{p_m q_n}. \tag{5.5}$$

In order to be able to use the mutual information $I_1(\xi, \eta)$ as a measure for the statistical dependencies among realizations of random variables, it has to show the following properties.

Theorem 5.1 (Properties of the Mutual Information $I_1(\xi, \eta)$)
The mutual information depicted in Equation (5.5) has the following properties:

1. *Symmetry:*

$$I_1(\xi, \eta) = I_1(\eta, \xi)$$

2. *Limitation:*

$$0 \leq I_1(\xi, \eta)$$

3. *Independency:*

$$I_1(\xi, \eta) = 0 \iff \xi \text{ and } \eta \text{ are statistical independent.}$$

4. Determination:

$$I_1(\xi,\eta) = H_1(\eta) \iff \eta \text{ is exclusively determined by } \xi,$$
$$I_1(\xi,\eta) = H_1(\xi) \iff \xi \text{ is exclusively determined by } \eta.$$

For a proof of the above properties see [Rén77].

Property 3 states, that if and only if no statistical dependency between both variables ξ and η exist, the mutual information disappears. Compared to the conventional coefficient of correlation ρ, the statistical independence of two random variables cannot be concluded from $\rho = 0$ for arbitrary distributed random variables.
Property 4 denotes, that the mutual information reaches its maximum if and only if $s_{m,n^*} = p_m$, for exactly one $n^* \in \{1, ..., N\}$ and hence $s_{m,n} = 0$ holds for all $n \neq n^*$. From this follows the functional dependency $\eta = f(\xi)$.

The mutual information $I_1(\xi,\eta)$ is apparently a reasonable measure for all statistical dependencies among random variables. The question is now, whether the properties of Theorem 5.1 hold as well for the general mutual information (GMI) $I_\alpha(\xi,\eta)$, with $\alpha > 0, \alpha \neq 1$. Unfortunately, this assumption is not met generally for arbitrary distributions S of the compound random variable (ξ,η).

Theorem 5.2 (Limitation of the GMI $I_\alpha(\xi,\eta)$)
Let S be the finite, discrete distribution of the compound random variable (ξ,η). For all $\alpha \geq 0, \alpha \neq 1$ a distribution S exists for which the quantity $I_\alpha(\xi,\eta)$ adopts negative values.
For an arbitrary distribution S, $I_\alpha(\xi,\eta) \geq 0$ holds only if $\alpha = 0$ or $\alpha = 1$.

A detailed proof of this theorem is given in [Rén77].

In this work, the general mutual information $I_2(\xi,\eta)$ is of particular interest. It can be estimated much more efficiently for time series of finite length than the mutual information $I_1(\xi,\eta)$ based on the Shannon-Entropy measure. An efficient estimation algorithm for $I_2(\xi,\eta)$ will be introduced in Sec. 5.3. Hence, the focus of interest is set on cases in which $I_2(\xi,\eta)$ can be utilized as a measure for statistical dependencies. The following theorem specifies the conditions under which $I_2(\xi,\eta)$ has the required properties.

Theorem 5.3 (Properties of the GMI $I_2(\xi, \eta)$)
Let $P = \{p_m\}_{m=1}^{M}, Q = \{q_n\}_{n=1}^{N}$ and $S = \{s_{m,n}\}_{m=1,n=1}^{M,N}$ be probability distributions of the random variables ξ, η and (ξ, η), respectively.
Let at least η be uniformly distributed, with $q_n = N^{-1}$ for $n = 1, ..., N$.
The general mutual information

$$I_2(\xi, \eta) = H_2(\xi) + H_2(\eta) - H_2((\xi, \eta)) \tag{5.6}$$

has the following properties:

1. *Symmetry:*
$$I_2(\xi, \eta) = I_2(\eta, \xi)$$
2. *Limitation:*
$$0 \leq I_2(\xi, \eta)$$
3. *Independency:*
$$I_2(\xi, \eta) = 0 \iff \xi \text{ and } \eta \text{ are statistical independent.}$$
4. *Determination:*
$$\begin{aligned} I_2(\xi, \eta) = H_2(\eta) &\iff \eta \text{ is solely determined by } \xi, \\ I_2(\xi, \eta) = H_2(\xi) &\iff \xi \text{ is solely determined by } \eta. \end{aligned} \tag{5.7}$$

The proof for this theorem is given in Appendix A.

For statistical independent random variables ξ and η it holds that $I_2(\xi, \eta) = 0$ even if η is arbitrarily distributed. For the converse to hold, the uniform distribution of at least η is essential, as expressed in Theorem 5.3.
As an immediate consequence of Equation 5.7, the general mutual information can be normalized to fit the interval $[0, 1]$, using:

$$0 \leq \frac{I_2(\xi, \eta)}{H_2(\eta)} \leq 1. \tag{5.8}$$

It will be shown in Section 7.2.3 how this normalized GMI can be used for a nonlinear analysis of real process data to obtain optimal data sets for neural network training. This normalized GMI can be used as an alternative

			I_0	I_1	I_2	C^2
η arbitrarily distrib.:						
"measure $\geqq 0$"			+	+	−	+
"measure $= 0$"	$\Rightarrow$	independence	−	+	−	−
"measure $= 0$"	$\Rightarrow$	uncorrelated	−	+	−	+
independence	$\Rightarrow$	"measure $= 0$"	−	+	+	+
uncorrelated	$\Rightarrow$	"measure $= 0$"	−	−	−	+
η uniformly distrib.:						
"measure $\geqq 0$"			+	+	+	+
"measure $= 0$"	$\Rightarrow$	independence	−	+	+	−
"measure $= 0$"	$\Rightarrow$	uncorrelated	−	+	+	+
independence	$\Rightarrow$	"measure $= 0$"	+	+	+	+
uncorrelated	$\Rightarrow$	"measure $= 0$"	−	−	−	+
+: true		−: false				

Table 5.1: *Comparison of the properties of various mutual information measures I_α between two random variables ξ and η with the properties of the squared coefficient of correlation C^2.*

to the coefficient of correlation, which describes a normalized covariance in the interval $[-1, 1]$. The major drawback of correlation analysis is its limitation to mere linear statistical dependencies for arbitrarily distributed random variables.

Table 5.1 has been extracted from [Pom98]. It compares the properties of various mutual information measures I_α with the properties of the squared coefficient of correlation C^2. The mutual information I_1 and I_2 are the only correlation measures that allow for the conclusion of statistical independence from a value of zero.
For an actual computation of the GMI, the requirement of a uniformly distributed random variable η might appear quite restrictive. We will see that this requirement is not as restrictive as it appears, since the estimation algorithm deals with preprocessed time series instead of the original data. In the following section, an efficient estimation algorithm for the GMI will be introduced.

5.3 Estimating the General Mutual Information I_2

In the previous section, the general mutual information $I_2(\xi, \eta)$ has been introduced from a theoretical viewpoint as a measure for linear and nonlinear statistical dependencies among random variables. Its employment is highly motivated by the existence of an efficient estimation algorithm for the second order Rényi-Entropy H_2. The basic ideas have been developed by TAKENS [Tak83]. Later GRASSBERGER [GSC91] estimated the fractal dimensions from finite time series of dynamic systems. These ideas have been combined for the estimation of the GMI by POMPE [Pom93]. The efficiency of this method is based on the fact that the GMI algorithm takes preprocessed time series instead of the original data. In the following, we will take a closer look at the more practical implementation aspects of the estimation algorithm. However, since the entire theoretical background goes far beyond the scope of this work, the interested reader is referred to [Pom97], [Pom98] and [PH95] for further reading.

5.3.1 Approximation of the Entropy Measure H_2

Let $\vec{\xi} := (\xi_1, ..., \xi_D)$ be a D-dimensional multivariate random variable and let $\vec{\varepsilon} := (\varepsilon_1, ..., \varepsilon_D)$ be a multidimensional vector defining the so called coarseness levels. With respect to ε_d, the variables ξ_d can be interpreted as discrete random variables $[\xi_d]_{\varepsilon_d}$, with $d = 1, ..., D$. The discrete random vector $\vec{\xi}$ can now be written as

$$[\vec{\xi}]_{\vec{\varepsilon}} = ([\xi_1]_{\varepsilon_1}, ..., [\xi_D]_{\varepsilon_D}),$$

with its according probability density distribution

$$\{p_m\} = \{p_{m_1 ... m_D}\}.$$

For the sake of generality, a separate quantization level ε_d is used for each random variable ξ_d. Hence, $p_m \equiv p_{m_1 ... m_D}$ is the probability of $\vec{\xi}$ adopting a value in a multidimensional box $B_m \equiv B_{m_1 ... m_D} \subseteq \mathbb{R}^D$. If $[\vec{\xi}]_{\vec{\varepsilon}}$ depicts a D-dimensional measurement vector, p_m can be estimated from its finite set of realizations

$$\{\vec{x}(m)\}_{m=1}^{K} = \{x_1(m), ..., x_D(m)\}_{m=1}^{K},$$

by employing

$$p_m \approx c_{\vec{\varepsilon},K}(m) = \frac{|\{k : \quad \| \vec{x}(m) - \vec{x}(k) \| < \frac{1}{2}\vec{\varepsilon}\}|}{K}, \quad k = 1, ..., K. \qquad (5.9)$$

Equation 5.9 depicts the well known method for estimating the probability densities from finite data sets, the so called naive density estimator. The variable $c_{\vec{\varepsilon},K}(m)$ is the relative frequency that arbitrary realization of the D-dimensional time series $\vec{x}(k)$ are within the D-dimensional interval defined by $\vec{\varepsilon}$. For time series of finite length, $c_{\vec{\varepsilon},K}(m)$ is an approximation for the probability p_m that a box B_m around a selected point $\vec{x}(m)$ contains an arbitrary realization of $\vec{x}(k)$. Figure 5.3 illustrates the determination of this relative frequencies for a one dimensional, time discrete time series of finite length K.
The quality of approximation depends on the smoothness of the underlying density function to be determined and also on the maximum ε_{max} of the corresponding box sizes. Assuming time series of infinite length, and an infinitesimal small maximum box size ε_{max}, the relative frequency $c_{\varepsilon_{max},K}(m)$ converges to its corresponding probability p_m.

$$p_m = \lim_{\varepsilon_{max} \to 0} \lim_{K \to \infty} c_{\varepsilon_{max},K}(m), \qquad \varepsilon_{max} = \max\{\varepsilon_D, ..., \varepsilon_1\}. \qquad (5.10)$$

Since data sets of finite length are considered, the limit in Equation 5.10 cannot be executed for the practical issues of an estimation algorithm. Hence, for time series of finite length, the relative frequencies $c_{\vec{\varepsilon},K}(m)$ will have to be used instead.

Figure 5.3 illustrates the determination of the required relative frequencies for a one dimensional time series of finite length. The absolute range of the time series is divided into equally spaced intervals of length ε_1. The relative frequency $c_{\varepsilon_1,K}(m_1)$ of a particular interval m_1 is then determined as the number of value in this interval with respect to the total length of the investigated time series.

Fig. 5.4 illustrates the estimation of the relative frequencies for a two dimensional signal $\vec{x}(k) \subseteq I\!R^2$. In the case of multidimensional random variables, different coarseness levels ε_d have to be used for each component.

Considering the Rényi-Entropies from Equation 5.4 and the estimated probabilities in Equation 5.9, POMPE [Pom97] presumably employs the

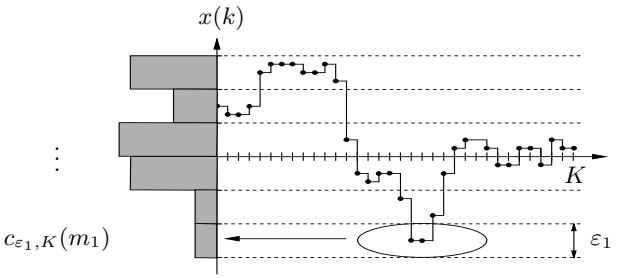


Figure 5.3: *Estimation of the relative frequencies $c_{\varepsilon_1,K}(m)$ for a one dimensional time series $x(k) \subseteq \mathbb{R}^1$.*

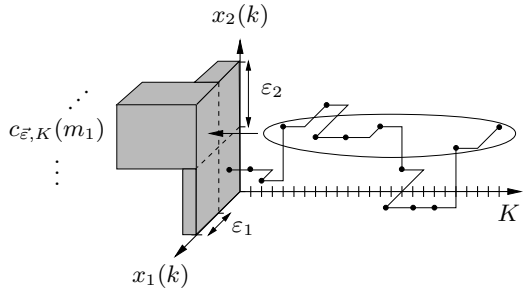


Figure 5.4: *Estimation of the relative frequencies $c_{\vec{\varepsilon},K}(m)$ of a two dimensional signal $\vec{x}(k) \subseteq \mathbb{R}^2$.*

following approximation:

$$\begin{aligned} \sum_m p_m^\alpha &\approx \frac{1}{K} \sum_m p_m^{\alpha-1}, \quad for \quad \alpha \geq 0, \alpha \neq 1 \\ p_m \log_2 p_m &\approx \frac{1}{K} \log_2 p_m, \qquad for \quad \alpha = 1. \end{aligned} \tag{5.11}$$

When using the terms on the right side of Equations 5.11, the entropy measures $H_\alpha(\vec{\xi})$ can be approximately derived with the following expressions:

$$H_\alpha(\vec{\xi}) \approx \begin{cases} \frac{1}{1-\alpha} log_2 \frac{1}{K} \sum_{m=1}^{K} c_{\vec{\varepsilon},K}^{\alpha-1}(m) & : \quad \alpha \geq 0, \alpha \neq 1 \\ -\frac{1}{K} \sum_{m=1}^{K} log_2 \, c_{\vec{\varepsilon},K}(m) & : \quad \alpha = 1. \end{cases} \tag{5.12}$$

For further investigation, only the entropy measure $H_2(\vec{\xi})$ will be considered. With respect to POMPE [Pom97], this measure turns out to be most suitable for the formulation of an efficient estimation algorithm.
Due to the finite length of the underlying time series, the desired entropy measure $H_2(\vec{\xi})$ can finally be approximated by the expression

$$H_2(\vec{\xi}) = \lim_{\varepsilon_{max} \to 0} \lim_{K \to \infty} \left(-log_2 \frac{1}{K} \sum_{m=1}^{K} c_{\vec{\varepsilon},K}(m) \right) \approx -log_2 \frac{1}{K} \sum_{m=1}^{K} c_{\vec{\varepsilon},K}(m). \tag{5.13}$$

In Equation 5.13, the term

$$C_{\vec{\varepsilon},K} = \frac{1}{K} \sum_{m=1}^{K} c_{\vec{\varepsilon},K}(m) \tag{5.14}$$

might be interpreted as a correlation integral of the underlying time series. It will be shown in Section 5.3.3 how this term can be very efficiently estimated. The algorithm which will be presented uses a binary representation of the underlying, possibly multidimensional, time series and appropriate elementary binary operations.

Considering another time series $\{\vec{y}(k)\}_{k=1}^{K}$, its correlation integral $C_{\vec{\varepsilon}_\star,K}$ and the integral $C_{(\vec{\varepsilon},\vec{\varepsilon}_\star),K}$ of the compound random variable $(\vec{\xi},\vec{\eta})$ can also be determined with the procedure presented above. With Equation 5.6 and Equation 5.13, a measure for the general mutual information $I_2(\vec{\xi},\vec{\eta})$ can thus be approximated using

$$I_{2_{(\vec{\varepsilon},\vec{\varepsilon}_\star)}}(\vec{\xi},\vec{\eta}) \approx -\log_2 C_{\vec{\varepsilon}_\star,K} - \log_2 C_{\vec{\varepsilon},K} + \log_2 C_{(\vec{\varepsilon},\vec{\varepsilon}_\star),K}. \tag{5.15}$$

The main task is now to compute these quantities from real time series of finite length. In the following, an efficient method for the computation of the required correlation integrals will be presented.

5.3.2 Ranking of the Time Series

For the interpretation of the quantity $I_2(\vec{\xi}, \vec{\eta})$ as a measure for the general mutual information, the realizations of all components of the random variable $\vec{\eta}$ have to be uniformly distributed over the unity interval $[0, 1]$. In order to meet the requirements of Theorem 5.3, only preprocessed time series are employed instead of the original raw data. When exemplary regarding a particular component η_i of a random variable $\vec{\eta}$ with arbitrarily distributed realizations, the transformation process is performed with its cumulative distribution function.

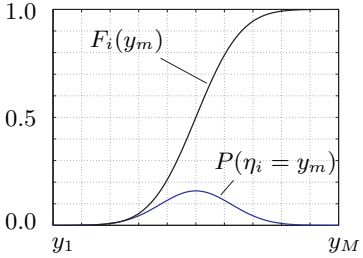


Figure 5.5: *Probability density function* $P(\eta_i = y_m)$ *and its cumulative distribution function* $F(y_m) = P(\eta_i \leq y_m)$.

Figure 5.5 depicts the probability density function $P(\eta_i = y_m)$ and the according cumulative distribution function $F_i(y_m)$ of the random variable η_i. This cumulative distribution function is used for the transformation of the arbitrarily distributed random variable $\vec{\eta_i}$ to a uniform distribution. All components of the multidimensional random variable $\vec{\eta}$ are successively transformed into a uniform distribution by mapping them through their according cumulative distribution function $F_i(\eta_i)$.
Figure 5.6 depicts the transformation of arbitrarily distributed real measurement data to uniform distribution. The probability density function of the original and the transformed signal is depicted at the right edge of the signal plots. When performing this operation, the dynamic characteristics of the original times series are maintained. In the end, the obtained signals are uniformly distributed over the interval $[0, 1]$.
However for practical aspects, this transformation can be realized more efficiently by computing the rank numbers of the underlying original time series [Pom98]. Equation 5.16 describes the ranking of an original time

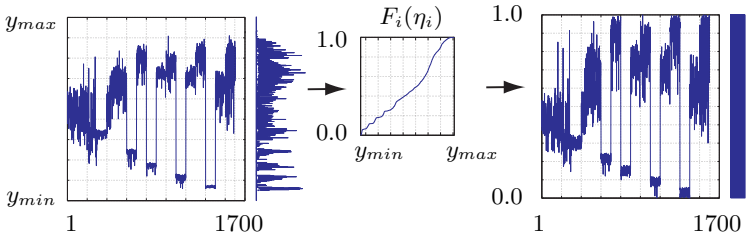


Figure 5.6: *Nonlinear transformation of an arbitrarily distributed signal to relative ranks with a uniform distribution.*

series to its absolute rank numbers $R_i(m)$ or alternatively to its relative ranks $r_i(m)$.

$$\begin{aligned} x_i(m) \mapsto R_i(m) &= |\{k : x_i(k) \leq x_i(m), \quad k = 1, ..., K\}|, \\ & \quad m = 1, ..., K \\ r_i(m) &= \frac{R_i(m)}{K} \end{aligned} \tag{5.16}$$

Using the transformation to absolute- or relative ranks, the characteristic of the obtained sequence is equivalent to the characteristic of the time series on the right-hand side of Fig. 5.6.
In the following presentation of a low level matrix calculus for the determination of the desired correlation integrals $C_{\vec{\varepsilon},K}$, only the relative ranks $r_i(m)$ will be considered.

5.3.3 Efficient Matrix Calculus at Bit Level

In order to formulate a fast estimation algorithm for the general mutual information $I_2(\vec{\xi},\vec{\eta})$, an efficient method for computing the correlation integrals in Equation 5.14 is required. For this purpose an appropriate low level matrix calculus is presented, which could be most efficiently implemented at bit level [Pom97].

Let $\{\vec{r}(m)\}_{m=1}^{K}$ be a (D+1)-dimensional sequence of relative rank numbers. The according ((D+1)×K)-dimensional rank matrix has the following form:

$$\{\vec{r}(m)\}_{m=1}^{K} = \begin{pmatrix} r_0(1) & r_0(2) & ... & r_0(K) \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ r_D(1) & r_D(2) & ... & r_D(K) \end{pmatrix}. \tag{5.17}$$

In order to compute the required correlation integral, all vectors $\{r_d(m)\}_{m=1}^{K}$, $d = 0, ..., D$ of the above matrix have to be considered.

The $(D+1)$ rank-distance matrices are defined by

$$\Delta_d = \begin{pmatrix} \delta_{d,(1,1)} & \delta_{d,(1,2)} & ... & \delta_{d,(1,K)} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \delta_{d,(K,1)} & \delta_{d,(K,2)} & ... & \delta_{d,(K,K)} \end{pmatrix}, \tag{5.18}$$

with

$$\delta_{d,(i,j)} = \parallel r_d(i) - r_d(j) \parallel, \quad i,j = 1, ..., K$$

are introduced for each $d = 0, ..., D$. The K^2 entries of matrix Δ_d represent the relative distances between all pairs of rank numbers in the particular sequence $\{r_d(m)\}_{m=1}^{K}$.

The matrix Δ_d has the following essential properties:

- Δ_d is symmetric with respect to its main diagonal,

$$\delta_{d,(i,j)} = \delta_{d,(j,i)}.$$

- The diagonal elements of Δ_d are all zero,

$$\delta_{d,(j,j)} = 0.$$

- Every row $\{\delta_{d,(i,j)}\}_{j=1}^{K}$ of Δ_d depicts a permutation of

$$\{R_d(m) - 1, R_d(m) - 2, ..., 1, 0, 1, K - R_d(m)\}.$$

- Every column $\{\delta_{d,(i,j)}\}_{i=1}^{K}$ of Δ_d depicts a permutation of

$$\{R_d(m) - 1, R_d(m) - 2, ..., 1, 0, 1, K - R_d(m)\}.$$

- For a particular sequence $\{R_d(m)\}_{m=1}^{K}$, $K!$ permutations exist. This implies the existence of $K!/2$ distinct matrices Δ_d, since the two sequences $\{R_d(m)\}_{m=1}^{K}$ and $\{1 + K - R_d(m)\}_{m=1}^{K}$ produce identical matrices

Furthermore, let ε_d be the coarseness level, with $0 < \varepsilon_d \ll 1$. The entries of a so called binary rank distance matrix

$$B_d = \begin{pmatrix} b_{d,(1,1)} & b_{d,(1,2)} & ... & b_{d,(1,K)} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ b_{d,(K,1)} & b_{d,(K,2)} & ... & b_{d,(K,K)} \end{pmatrix}. \tag{5.19}$$

are computed according to the rule

$$b_{d,(i,j)} := \begin{cases} 1 & : \quad \delta_{d,(i,j)} < \varepsilon_d \\ 0 & : \quad \text{else.} \end{cases} \tag{5.20}$$

The resulting binary matrix B_d has the following properties:

- B_d is symmetric with respect to its main diagonal,

$$b_{d,(i,j)} = b_{d,(j,i)}. \tag{5.21}$$

- The diagonal elements of B_d are all one,

$$b_{d,(j,j)} = 1. \tag{5.22}$$

- For $r_d(m) \geq \frac{\varepsilon_d}{2}$, the weight[5] of the m-th row of B_d is at most $K\,(\varepsilon_d - \frac{1}{K})$. The same holds for the m-th column of B_d.

In consequence of the symmetry of B_d, only the upper triangular matrix is considered for further calculations.

The left-hand side of Figure 5.7 shows the binary rank distance matrix of the signal depicted in Fig. 5.6. Since the diagonal elements are permanently equal to one, they can also be factored out for the sake of performance.

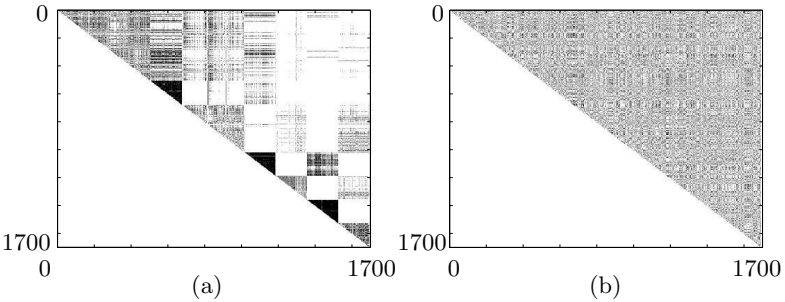


Figure 5.7: *(a) Binary rank distance matrix B_d of the signal depicted in Fig. 5.6. (b) Binary matrix of a white noise process, showing no statistic dependencies. In this image representation, the binary ones are depicted as black dots.*

The binary matrices still contain all information about the statistic dependencies[6] of the considered time series, with respect to a predefined coarseness level ε_d. In the case of a $D+1$ time series, the desired binary rank distance matrix is computed as the conjunction of the binary matrices for each dimensionality. The procedure for the determination of the coarseness levels is described in Section 5.3.4.

[5] The weight of a binary vector or matrix is defined as the sum of binary ones.

[6] Binary rank distance matrices of signals without statistic dependencies are equally gray (Fig. 5.7b). The gray level increases if the coarseness level ε_d decreases.

Definition 5.1 (Conjunction of Binary Matrices)
Let $B_0, ..., B_D$, $D \in \mathbb{N}$ be binary matrices as introduced in Equation 5.19. The conjunction of multiple binary matrices is defined as follows:

$$\bigwedge_{d=0}^{D} B_d = (b_{0,(i,j)} \wedge \ ... \ \wedge b_{D,(i,j)}), \quad i,j = 1, ..., K. \tag{5.23}$$

The question of interest is how this binary matrix calculus can be employed to determine the desired correlation integral $C_{\vec{\varepsilon},K}$ in Equation 5.14. This question can be answered by taking a closer look at the terms $c_{\vec{\varepsilon},K}(m)$ in Equation 5.14 and the according definitions in Equation 5.9. The correlation integral $C_{\vec{\varepsilon},K}$ can thus be written as

$$C_{\vec{\varepsilon},K} \quad = \quad \frac{1}{K^2} \sum_{m=1}^{K} |\{k : \quad \| \vec{r}(m) - \vec{r}(k) \| < \frac{1}{2} \vec{\varepsilon}\}|, \tag{5.24}$$

with $k, m = 1, ..., K$. The vectors $\vec{r}(m)$ depict the relative ranks series.

On the one hand, the correlation integral can be computed as the sum of the presented set cardinalities with respect to the squared length of the underlying time series. On the other hand this sum is equal to the weight of the corresponding binary matrix which has been obtained using Equation 5.20. For a time series of dimension $D+1$ however, a prior conjunction of all component binary matrices has to be performed.

Hence, the correlation integral in Equation 5.14 can be obtained as the relative weight of the resulting binary matrix with respect to the squared length of the underlying time series.

$$C_{\vec{\varepsilon},K} = \frac{1}{K^2} \sum_{i,j=1}^{K} \left(\bigwedge_{d=0}^{D} B_d \right) = \frac{1}{K} + \frac{2}{K^2} \sum_{i<j} \left(\bigwedge_{d=0}^{D} B_d \right). \tag{5.25}$$

Finally, the approximation for the entropy measure $H_2(\xi)$ of a (D+1)-dimensional random variable is found to be

$$H_2(\vec{\xi}) \approx -\log_2(C_{\vec{\varepsilon},K}) = -\log_2 \left(\frac{1}{K} + \frac{2}{K^2} \sum_{i<j} \left(\bigwedge_{d=0}^{D} B_d \right) \right). \tag{5.26}$$

Since this method is mainly based upon comparisons and logical operation at bit level, the computation of the above entropy measure could

be most efficiently implemented on dedicated hardware. With respect to cost and performance, the combination of Field Programmable Gate Arrays (FPGA) and Digital Signal Processors (DSP) might be employed for this task. Due to some inconsistencies found in literature, the algorithmic complexity of the presented method is investigated in the section 5.4.

5.3.4 Determination of the Coarseness Level

An important issue, when employing the GMI estimation algorithm on the basis of the binary matrix calculus described in Section 5.3.3, is the determination of the dimension specific estimation parameters $\vec{\varepsilon}$. In Section 5.3.1, the vector $\vec{\varepsilon}$ has been defined to contain the coarseness levels of the entropy approximation algorithm. As described in [BDSL91], various approaches for the determination of the coarseness levels can be found in literature. Besides all that, a feasible alternative approach for the determination of this important estimation parameters has been developed in this work and will be presented in the following.

As depicted in Equation 5.9, the values of the coarseness level vector $\vec{\varepsilon}$ describe a D-dimensional box around a selected point $\vec{x}(m)$. When considering Equation 5.10, the multidimensional box can also be defined as $\vec{\varepsilon} = (\varepsilon_1, ..., \varepsilon_D)$, with $\varepsilon = \varepsilon_1 = ... = \varepsilon_D$, without loss of generality. The edge ε of this multidimensional cube is now determined in such a way, that the binary matrix for the compound variabel $(\vec{\xi}, \vec{\eta})$ contains an average of Kp binary ones with respect to the length K of the time series.

Hence, the concrete value of ε is determined iteratively through nested intervals with the stopping criterion

$$H_2((\vec{\xi}, \vec{\eta}), \varepsilon) = -\log_2(p). \tag{5.27}$$

In the case of convergence, this yields in conjunction with Equation 5.13 to the following:

$$H_2((\vec{\xi}, \vec{\eta}), \varepsilon) = -\log_2(p) = -log_2 \frac{1}{K} \sum_{m=1}^{K} c_{\vec{\varepsilon},K}(m)$$

$$\begin{aligned} \Leftrightarrow \quad p &= \frac{1}{K}\sum_{m=1}^{K} c_{\vec{\varepsilon},K}(m) \\ \Leftrightarrow \quad K \cdot p &= \frac{1}{K}\sum_{m=1}^{K} |\{k : \| \vec{x}(m) - \vec{x}(k) \| < \frac{1}{2}\vec{\varepsilon}\}|, \end{aligned} \tag{5.28}$$

where $\vec{\varepsilon} = (\varepsilon_1, ..., \varepsilon_d)$ with $\varepsilon = \varepsilon_1 = ... = \varepsilon_D$.

The right side of the above equation describes the average number of ones in the according binary matrix. We can see clearly that, at the end of the iteration procedure, the average number of binary ones is equal to the fraction p with respect to the total length K of the time series.

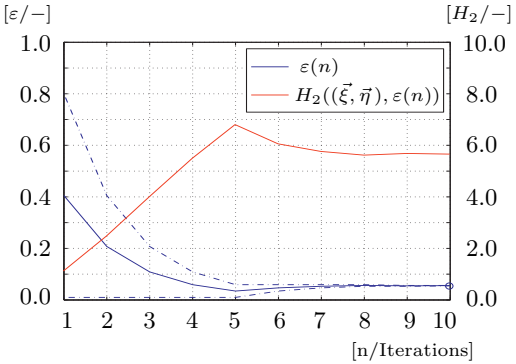


Figure 5.8: *Values of the coarseness level $\varepsilon(n)$ and the according entropy $H_2((\vec{\xi}, \vec{\eta}), \varepsilon(n))$ during iteration. The dashed lines represent the boundaries of the contractive interval for the search of ε, with $p = 0.02$.*

Figure 5.8 depicts the values of the coarseness level $\varepsilon(n)$ and the according entropy $H_2((\vec{\xi}, \vec{\eta}), \varepsilon(n))$ during the course of the iteration process. The dashed lines represent the contractive interval for the search of ε with $p = 0.02$. After ten iterations, the value of ε fulfills the stopping criterion in Equation 5.27. At this point the iteration process is terminated and the

current value for ε is used for further computations of the mutual information.
Finally, one might raise the question how the value of the constant p is determined anyway. The constant p has been empirically determined through various experiments. The best results are obtained for values $0.02 \leq p \leq 0.05$.

5.4 Algorithmic Complexity of the Estimation Procedure

One important point, with respect to an efficient implementation of the above estimation method for the entropy measure $H_2(\cdot)$ and thus for the mutual information $I_2(\xi, \eta)$, is the question of its algorithmic complexity. Since the required amount of memory depends on the machine representation of the investigated time series, we are only interested in the complexity class of the computational costs as a function of the length K of the investigated time series.

In general, the mutual information has to be determined for a D-dimensional input sequence

$$\{\vec{x}(m)\}_{m=1}^{K} = \begin{pmatrix} x_1(m) \\ \vdots \\ x_D(m) \end{pmatrix}$$

and a G-dimensional output sequence

$$\{\vec{y}(m)\}_{m=1}^{K} = \begin{pmatrix} y_1(m) \\ \vdots \\ y_G(m) \end{pmatrix}.$$

The number of subtractions and comparisons to be carried out for obtaining the binary matrix of one particular time series is

$$\underbrace{\frac{K(K-2)}{2}}_{subtractions} + \underbrace{\frac{K(K-2)}{2}}_{comparisons}.$$

For the determination of all binary matrices B_d and B_g, with $d = 1, ..., D$ and $g = 1, ..., G$ subtractions and comparisons are necessary.

$$\underbrace{(D+G)\frac{K(K-2)}{2}}_{subtractions} + \underbrace{(D+G)\frac{K(K-2)}{2}}_{comparisons} = (D+G)K(K-2) \in O(K^2)$$

The number of logical AND operations to be carried out for obtaining the matrices $\bigwedge_{d=1}^{D} B_d$, $\bigwedge_{g=1}^{G} B_g$ and $(\bigwedge_{d=1}^{D} B_d) \wedge (\bigwedge_{g=1}^{G} B_g)$ is

$$\begin{aligned} & \underbrace{D\frac{K(K-2)}{2}}_{\bigwedge_{d=1}^{D} B_d} + \underbrace{G\frac{K(K-2)}{2}}_{(\bigwedge_{g=1}^{G} B_g} + \underbrace{\frac{K(K-2)}{2}}_{(\bigwedge_{d=1}^{D} B_d)\wedge(\bigwedge_{g=1}^{G} B_g)} \\ = \quad & (D+G+1)\frac{K(K-2)}{2} \in O(K^2). \end{aligned}$$

The number of counter increments in order to determine the weights of the binary matrices $\bigwedge_{d=1}^{D} B_d$, $\bigwedge_{g=1}^{G} B_g$ and $(\bigwedge_{d=1}^{D} B_d) \wedge (\bigwedge_{g=1}^{G} B_g)$ is

$$\begin{aligned} & \underbrace{\frac{K(K-2)}{2}}_{(\bigwedge_{d=1}^{D} B_d)} + \underbrace{\frac{K(K-2)}{2}}_{(\bigwedge_{g=0}^{G} B_g)} + \underbrace{\frac{K(K-2)}{2}}_{(\bigwedge_{d=1}^{D} B_d)\wedge(\bigwedge_{g=1}^{G} B_g)} \\ = \quad & \frac{3}{2}K(K-2) \in O(K^2). \end{aligned}$$

The transformation of the arbitrarily distributed time series to its corresponding series of rank numbers is realized as sorting operation. Since the Quicksort algorithm has been used, the complexity class for obtaining the ranked time series is $O(K \log K)$.

Finally, the algorithmic complexity class of the entire estimation algorithm is therefore $O(K^2)$. Considering the representation of the time series as sequences of absolute rank numbers, this algorithm could be most efficiently implemented at bit level on dedicated hardware, as mentioned above.

5.5 Analysis of Nonlinear Dynamic Process Inputs with the GMI

In this section, the GMI will be employed for the first time to identify the relevant inputs of a nonlinear dynamic process. In order to demonstrate the capability of the GMI to identify implicit nonlinear structural- and temporal dependencies, the input-/output behavior of a nonlinear dynamic process will be analyzed. Unlike the investigation of the characteristics of real measurement data, the properties of a deterministic dynamic process can be specifically tailored to have the desired dependencies, which will be revealed by the GMI.

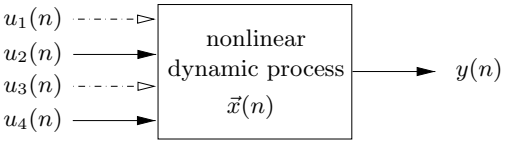


Figure 5.9: *General description of the deterministic nonlinear dynamic process, defined in Equations 5.29.*

$$
\begin{aligned}
x_1(n+1) &= x_1(n) - 0.5\,x_2(n) + 0.8\,u_2^2(n-k_2) + u_4^2(n-k_4), \\
x_2(n+1) &= \frac{x_1(n)}{(1+0.1\,x_2^2(n))} + 0.8\,u_2^2(n-k_2) + u_4^2(n-k_4), \\
y(n) &= 1.8\tanh\left(0.32\,x_1(n)\right) - 0.63\,, \qquad (5.29)
\end{aligned}
$$

Figure 5.9 depicts the general description of the nonlinear dynamic process, defined in Equations 5.29. The process itself is designed to have two internal states, which are also nonlinearly connected to each other.
It can be observed in the graphical- and in the analytical representation, that only the variables u_2 and u_4 are used as inputs to the dynamic process, while variables u_1 and u_3 are redundant.
Since the GMI is also supposed to identify the implicit temporal dependencies, the signals of variables u_2 and u_4 have been delayed by $k_2 = 10$ and $k_4 = 20$ time steps, respectively.

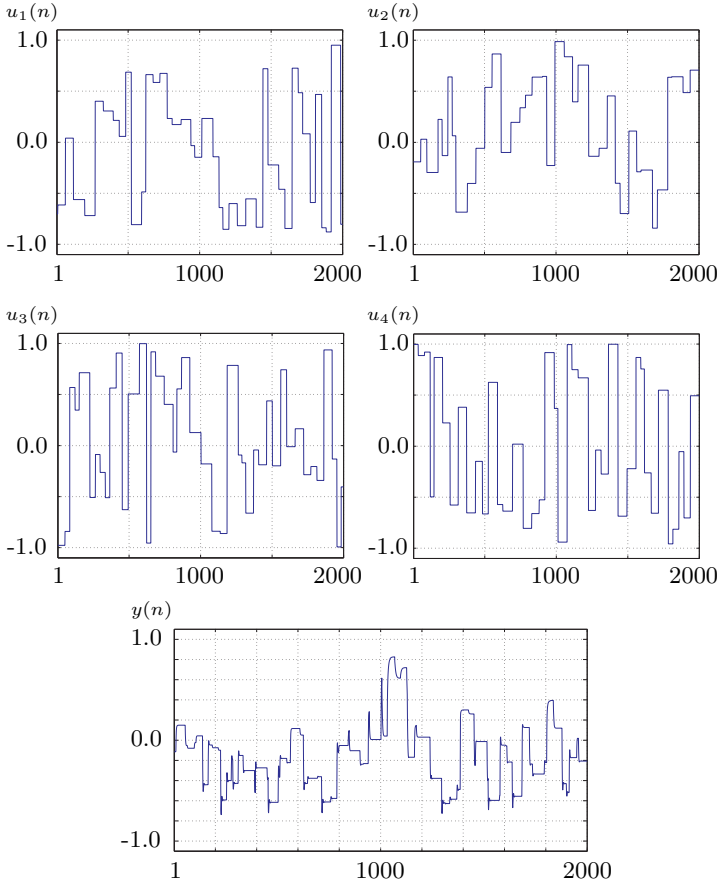


Figure 5.10: *Response of the nonlinear dynamic system to square-pulse input sequences. The variables u_2 and u_4 are used as inputs to the dynamic process, while u_1 and u_3 are redundant.*

The response of nonlinear, dynamic system to square-pulse input sequences is depicted in Figure 5.10. Since the GMI should give information about

the implicit structural- and temporal dependencies, we pretend to have no further a priori knowledge about this data or the structure of the underlying dynamic process.

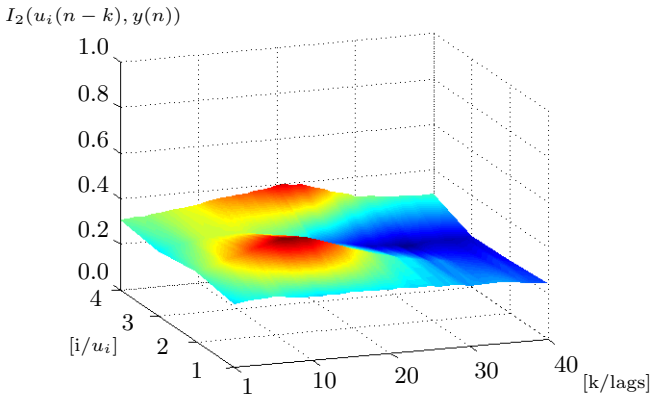


Figure 5.11: *The GMI as a function of the input variable u_i and a specific time lag k.*

In Figure 5.11, the result of the GMI analysis of the available process data is depicted. It can clearly be observed, that the GMI values adopt only two local maxima throughout the entire range.
The first maximum can be located for variable u_2 at time lag $k = 10$ and the second can be found for variable u_4 and time lag $k = 20$. However, for all other variables and time lags, no significant change of the GMI values can be observed. This results correspond exactly to the definition of the nonlinear dynamic process in Equations 5.29.

Finally, by assuming that no a priori information about the internal structure of the dynamic process existed, it has to be concluded from the above results to omit the variables u_1 and u_3, while retaining the variables u_2 and u_4. Furthermore, when concerning the temporal aspect of the accomplished GMI analysis, one should also conclude to employ the delayed input sequences $u_2(n-10)$ and $u_4(n-20)$ for further process identification and modelling purposes.

In this context, another question of particular interest is how the GMI, as a measure of statistical dependency, compares to the well known correlation analysis. As a matter of fact, the GMI already revealed exactly the structural- and temporal dependencies with which the nonlinear dynamic process had been designed.

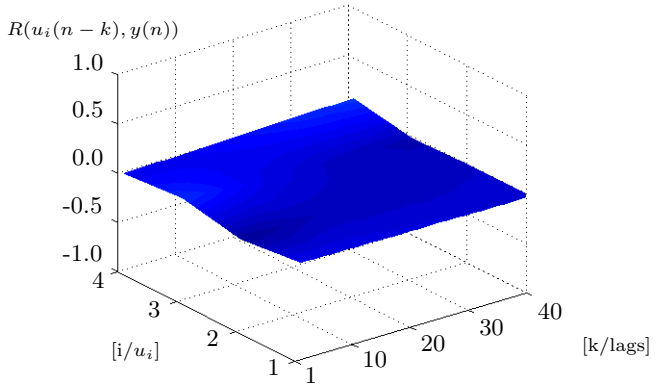


Figure 5.12: *Spearman's rank correlation coefficients as a function of the input variable u_i and a specific time lag k.*

Figure 5.12 depicts the result of the correlation analysis of the dynamic process data from above. Without exception, it can be observed that the correlation coefficients adopt values around zero for all combinations of variables and time lags. When considering the properties of the mutual information and the coefficient of correlation in Table 5.1, no conclusions with respect to statistical dependency can be drawn from this outcome at all. Hence, the correlation analysis would fail at providing the structural- and temporal information, which is required for successful system identification.

In the following, a neural system identification of the nonlinear dynamic process described in Equations 5.29 will be conducted. It will be demonstrated, how a Recurrent Multilayer Perceptron does perform with and without the knowledge about the intrinsic properties of the dynamic process, selected by the GMI. The Extended Kalman Filter, as described in

Section 4.3.4, will be employed as training algorithm for the recurrent neural structures. In both cases, the learning rate has been set to $\eta = 0.001$.

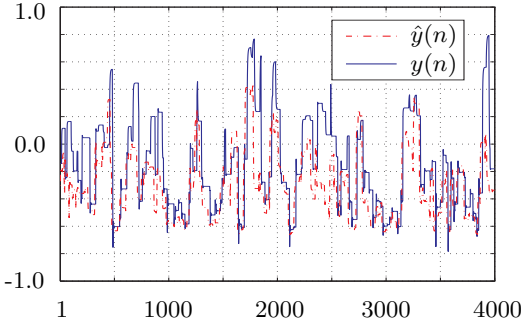


Figure 5.13: *Neural identification result of a Recurrent Multilayer Perceptron RMLP-4-4-2-1 with Extended Kalman Filter training and no prior input variable selection.*

Figure 5.13 depicts the neural identification result obtained by a RMLP with Extended Kalman Filter training. The network estimates are denoted by $\hat{y}(n)$, whereas the desired target output is referred to as $< (n)$. In this case, all available input sequences without any time delay have been employed for training. After several identification runs, this result had an average squared error $ASE = 0.801$ with respect to the depicted test set of 4000 patterns.
It can be observed, that the neural structure could only adapt to a certain degree to the behavior of the process. In some instances, the RMLP does respond poorly, in other instances it does not respond at all, in the required way.

However, if the findings of the GMI analysis are employed, the neural identification result becomes quite accurate.

Figure 5.14 depicts the neural identification results after the removal of the redundant variables u_1 and u_3. In this case, the obtained time delays have been employed for the remaining input variables u_2 and u_4.
The test set had an average squared error of $ASE = 0.002$. It can be observed that the quality of the identification significantly improved after the

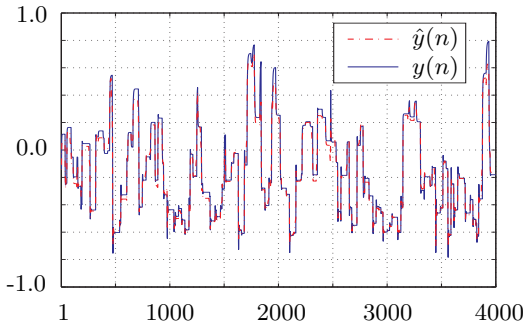


Figure 5.14: *Neural identification result of a Recurrent Multilayer Perceptron RMLP-2-4-2-1 with Extended Kalman Filter training. The GMI has been employed for prior input variable selection.*

redundant variables have been removed and the correct time delays were applied.

In this section, we have employed the GMI for the analysis of nonlinear dynamic process inputs. It turned out that the GMI is capable of revealing the implicit nonlinear structural- and temporal dependencies without any a priori knowledge about the underlying process.
However, if the intention is to employ the GMI for the analysis of real technical systems, defective measurement setup might provide low-value data. In other words, the acquired data might contain missing values due to unreliable measurement devices. This constellation will be investigated in the next section.

5.6 GMI-Analysis of Nonlinear Process Data Sets with Missing Values

In this section, we will investigate how the GMI can cope with data containing missing values. For this purpose, the nonlinear dynamic process data in Section 5.5 is artificially modified to contain a certain percentage

of missing values. The missing values will be uniformly distributed over the input- and the output-sequences. In our case, the missing values in the data set are represented as *NaN*, i.e. the IEEE arithmetic representation for *Not-a-Number*.

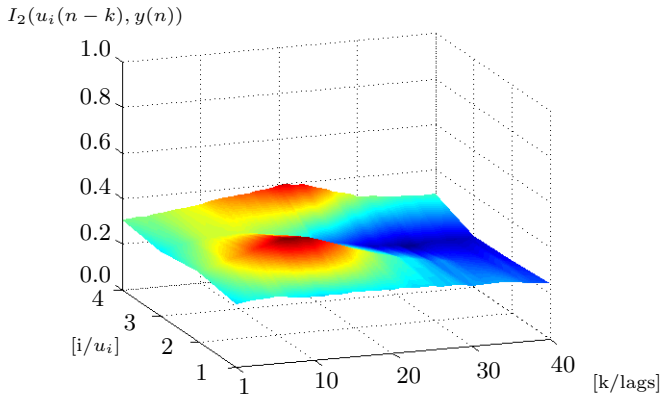


Figure 5.15: *The GMI as a function of the input variable u_i and a specific time lag k with no missing values.*

In Figure 5.15, the results of the GMI analysis with no missing values is depicted. As already presented in the Section 5.5, it can be clearly observed that the GMI values adopt two local maxima. The locations of these maxima correspond perfectly to the definition of the nonlinear dynamic process in Equations 5.29. However, in the next step we will investigate the effect of missing values on the outcome of the GMI analysis.

Figure 5.16 depicts the analysis outcome when 10% of the data set have been randomly labeled as missing values. It can be observed that the two local maxima are still present at the correct locations. Although the height of both maxima are slightly reduced, the temporal and structural dependencies can still be clearly deduced from the analysis result.
This situation changes completely if the data set contains 20% missing values as shown in Figure 5.17. Here, the GMI is depicted as a function of the input variable u_i and a specific time lag k. Although the maxima, indicating the specific input variables and its according time lags, can be assumed to be present, they are now unspecific and of diffuse shape. For instance,

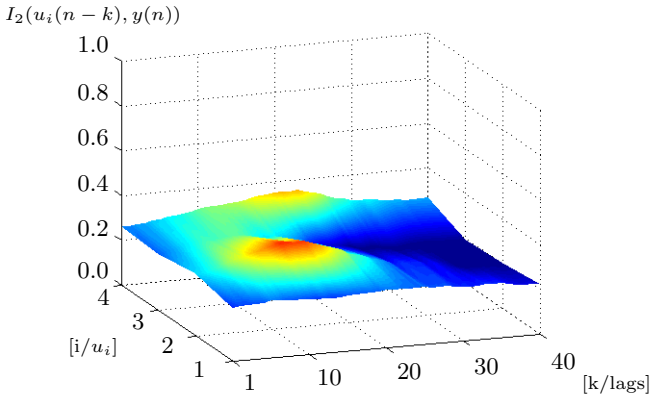


Figure 5.16: *The GMI as a function of the input variable u_i and a specific time lag k with 10% missing values. The two local maxima are still located at the correct position.*

the time lag of variable u_4 cannot be determined exactly from the analysis outcome. It can only be assumed to lie between 10 and 30 time lags. Hence, no clear statement concerning the temporal dependencies between u_4 and y can be concluded in this case.
In this section, the capability of the GMI to cope with missing values has been investigated. For this purpose, the nonlinear dynamic process data in Section 5.5 has been modified to contain missing values. The new data sets contained 0%, 10% and 20% uniformly distributed missing values, respectively. The modified data sets have been analyzed without the prior employment of missing-data procedures.
It turned out that, the GMI could be calculated without problems even though missing values were present in the data set. In the present case, the GMI could even cope with up to 10% missing values. In other words, the structural- and temporal dependencies of the underlying nonlinear, dynamic process could be revealed even if 10% of the analyzed data set content were missing values.
In order to understand why the GMI can cope with up to 10% missing values, we have to take a closer look at Algorithm B.2. This algorithm takes arbitrarily distributed time series and transforms them into uniformly distributed sequences of relative rank numbers. According to the employed

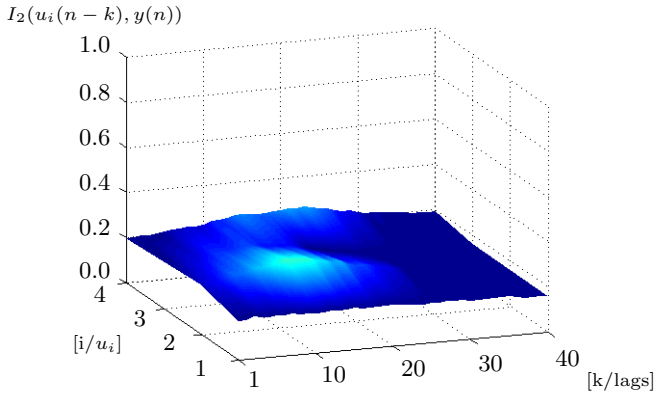


Figure 5.17: *The GMI as a function of the input variable u_i and a specific time lag k with 20% missing values. The two local maxima are now unspecific and of diffuse shape.*

analysis software, the missing values are represented as dedicated entries. In our case, the missing values in the data set are represented as *NaN*, which is the IEEE arithmetic representation for *Not-a-Number*. Since the core of this transformation process is an implementation variant of a standard sorting function, the set of missing values is always assigned to the set of highest values in the transformed sequence. Hence, when computing the GMI these values act as mere constant inputs with no functional dependency whatsoever and do not contribute to the final result.
The benefit of this property of the GMI is its intrinsic robustness against missing values. In fact, if no preprocessing with respect to missing-data is done, the GMI is able to cope with a rather large amount of missing values. However, since there is no benefit without a drawback, the quality of the GMI results deteriorates rapidly with an increasing percentage of persistent missing values. Hence, in cases with more than approximately 10% missing values, the observation time should be increased for the compensation of the missing data. If an extension of the observation time is not feasible, the prior employment of dedicated missing-data procedures is strongly indicated before the computation of the GMI.

6 Optimal Feature Selection with the GMI

The classic supervised learning of neural structures, for instance, relies on a training set comprised of input patterns, which are the realization of the feature vector, and the according output patterns. After successful learning, the neural structure is expected to serve as an acceptable model of reality which represents the behavior of the underlying mapping process between input- and output samples. Thus, the quality and reliability of the derived neural model is determined by the quality of the neural training set. This is in turn highly dependent on the amount of information that is inherent in the selected input variables which comprise the training set. One might assume theoretically, that more input variables or features should also contain a higher amount of information and thus yield more accurate models. However, reality provides us with many reasons why this is not true in general [KS96].

First of all, the time requirements for neural algorithms grow dramatically with the number of variables involved. In Section 7.2, it will be demonstrated how the general mutual information is used for feature selection and thus for dimension reduction of the neural input vector. In this well-illustrated case, the focus was to meet the predefined timing limits while performing real-time neural signal processing in an automotive environment with hard real-time requirements.
Furthermore, most learning algorithms can be viewed as performing a form of biased estimation of the output variable distributions given a set of features. If a large number of features is involved, the distributions are very complex and of considerably high dimension. Unfortunately, in the real world, we are always faced with the problem of limited data sources. This fact makes it very difficult to obtain good estimates for the many probabilistic parameters. This fact is also closely associated with the so called *curse of dimensionality*, mentioned earlier in Chapter 3. It states the expo-

nential growth of hyper-volume as a function of the input dimensionality and thus the ever-growing amount of required data to cover it.
Finally and most important, irrelevant or redundant features could cause severe problems in this context. They may confuse the derivation of the model by obscuring the information provided by a small set of truly relevant features for the task at hand. Although, if all required information is readily available, this constellation might yield essentially poor models.

The general mutual information, which represents a thoroughly investigated concept of information theory, has been used as the central point for the development of the feature selection method proposed in Section 6.2.3 and 6.3. The formulation of this feature selection strategy as so called greedy algorithm, will be justified in Section 6.1.
The new point in this context is to transfer Shannon's abstract model of a symmetric and noise contaminated communication channel to the problem of feature selection. The leading thought is the separation of important from unimportant features by measuring the abstract flow of information between input- *and* output variables directly with the GMI. While the PCA and the ICA in Chapter 3 inherently imply a linear model for the dependence of the input variables among each other, the GMI goes beyond this assumptions. Different from the PCA or the ICA, the GMI is capable to detect the *linear*- as well as the *nonlinear* dependencies between the input *and* the output variables. This small but rather significant difference qualifies the GMI as a versatile method for feature selection, as we will see in Chapter 7.

6.1 Theoretical Framework of Optimal Feature Selection

This section is dedicated to unveil the theoretical aspects which are behind the claim of an optimal feature selection.
In the first part of this section, the theoretical framework for the backward elimination strategy is presented. It employs the concept of the so called Markov Blankets, as provided in [KS96], to prove the efficacy of the backward elimination algorithm. However, the second part of this section will be concerned with the forward selection strategy. In order to justify the formulation of this selection strategy as a greedy algorithm, its monotone convergence property will be investigated.

6.1.1 Effectiveness Considerations of Backward Elimination

In the sense of time series analysis, a data instance is described as an assignment of values $\vec{f} = (f_1, ..., f_n)$ to a vector of random variables or features $\vec{F} = (F_1, ..., F_n)$. It is assumed, that each data instance is drawn independently according to some probability distribution over the feature space. The assignment of values $\vec{f}$ to the features $\vec{F}$ are described through the probability distribution $P(\vec{F} = \vec{f})$.
A classifier maps the above mentioned data instances to a number of possible classes $C = \{c_1, ..., c_l\}$. Optimistically, the feature vector $\vec{F}$ will fully determine the mapping from feature space to class space. Unfortunately, this is rarely the case. Since we usually do not have the right feature set to make this a deterministic decision, a probability distribution is used to model this classification function. The conditional probability $P(C = c_k \mid \vec{F} = \vec{f})$ describes the classification function which associates the data instances to the according classes.

Let us now consider the effect of feature space reduction on the distribution $P(C = c_k \mid \vec{F} = \vec{f})$. Let the components in a reduced feature vector $\vec{G}$ be some subset of the components in $\vec{F}$ and let $\vec{f}_{\vec{G}}$ denote the realizations of the variables of this new feature vector. In the original feature space, the distribution describing the classification function is $P(C = c_k \mid \vec{F} = \vec{f})$. In the reduced feature space, this distribution becomes $P(C = c_k \mid \vec{G} = \vec{f}_G)$. From a theoretic viewpoint, the goal of dimension reduction or feature selection is on the one hand to select $\vec{G}$ in such a way that the above distributions are as close as possible. On the other hand, the dimension of $\vec{G}$ has to be as small as possible and as large as necessary to solve the task at hand.

In this theoretic framework, the cross-entropy, also known as the KL-distance [KL51] is considered as the distance measure between the two distributions. Thus, the dimension reduction can be seen as identifying a lower dimensional feature vector $\vec{G}$, which causes the least loss of information when considering the original feature vector $\vec{F}$ as the baseline.

The previously mentioned the cross-entropy as a distance measure between two distributions is defined to be

$$\delta_G(\vec{f}) = \sum_{x \in \Omega} \mu(x) \log \frac{\mu(x)}{\sigma(x)}, \tag{6.1}$$

where $\mu = P(C = c_k \mid \vec{F} = \vec{f})$ and $\sigma = P(C = c_k \mid \vec{G} = \vec{f}_G)$. The set of elementary events, from which the classes are drawn, is denoted by Ω.
In order to compare one feature vector $\vec{G}$ to another, we must first integrate the values $\delta_G(\vec{f})$ over the different data realizations $\vec{f}$. Unfortunately, some vectors $\vec{f}$ are more likely to occur than others. To avoid making larger mistakes in certain cases, the distance measure is weighted with this probability. The new feature vector $\vec{G}$ is then obtained by minimizing the expression

$$\Delta_G = \sum_{\vec{f}} P(\vec{f})\, \delta_G(\vec{f}). \tag{6.2}$$

Clearly, the feature vector that minimizes this quantity is simply $\vec{F}$. This fact suggests the use of a so called backward elimination strategy, which starts out with all available features $\vec{F}$. In each iteration, a specific feature F_i is eliminated that allows to stay as close to the original distribution as possible. The fact, that once a feature has been eliminated, there is no need to reconsider it again in later iterations, is shown with the concept of Markov Blankets. As a consequence of this, the backward elimination strategy can be formulated as a greedy algorithm without loss of generality. A detailed outline of the backward elimination strategy is given in pseudo-code in Appendix B.2.
However, the computation expenses of the Δ_G grow exponentially in the number of features and available data instances. Furthermore, the iteratively obtained distribution $P(C \mid \vec{G} = \vec{f}_G)$ cannot be compared directly to the baseline distribution $P(C \mid \vec{F} = \vec{f})$, since it is usually not available. This fact renders the backward elimination approach infeasible for most high-dimensional technical problems e.g. as the one described in Section 7.2.3.

Definition 6.1 (Conditional Independence)
Let A, B and X be arbitrarily distributed random variables. The two random variables A and B are defined to be conditionally independent, if $P(A = a \mid X = x, B = b) = P(A = a \mid X = x)$.

Intuitively, features that are conditionally independent provide the least additional information. This fact results in a small increase in Δ. Thus, a conditionally independent feature F_i can be removed without increasing the distance from the baseline distribution.
Since it is impractical to test for conditional independence, a new formulation of the problem, employing the concept of so called *Markov Blankets*, points the way to a possible solution.

Definition 6.2 (Markov Blanket)
Let F and C be a set of features and a set of classes, respectively. Furthermore, let $M = F \setminus \{F_i\}$ be some set of features which does not contain F_i. The set M is a Markov Blanket for F_i, if F_i is conditionally independent of $(F \cup C) \setminus M$.

The Markov Blanket condition is stronger than the formulation of conditional independence. It requires that a Markov Blanket M subsumes not only the information that F_i has about a class C, but also of all other features. The intention of the backward selection process is to remove those features for which a Markov Blanket can be found within the set of remaining features. It can be shown, that features judged as unnecessary based on this criterion remain in fact unnecessary during the rest of the selection process.
Assume that a feature F_i has been removed on the basis of an existing Markov Blanket M. In some later iteration, another feature $F_j \in M$ will be removed. In general, the removal of F_j might now render F_i relevant again. The following theorem states that this is not true.

Theorem 6.1 (Optimal Backward Elimination)
Let G be a set of features. If $F_i \notin G$ and $F_j \in G$, $i \neq j$ and both have Markov Blankets in G. Then F_i has also a Markov Blanket in $G \setminus \{F_j\}$.

The proof of this theorem is based on the basic independence of probability distributions, as described in [Pea88]. For a detailed description of this proof, the reader is primarily referred to [KS96].

Theorem 6.1 delivers a very important statement concerning the optimal backward elimination strategy. A feature, that has been considered irrelevant and removed on the basis of an existing Markov Blanket, does not have to be reconsidered again. Thus, the Markov Blanket criterion only removes those features that are truly unnecessary. Furthermore, this concept removes all attributes that are irrelevant for the present task and it removes all redundant attributes which are already explained through

other attributes of the feature set. Anyway, the concept of Markov Blankets has been introduced in [KS96] against the background of the backward elimination strategy to prove its effectiveness. In this context, one major drawback exists. In the rather realistic case, where some features obscure the information provided by some other features, one would start the elimination process with the full feature set. The elimination process is proved to deliver optimal decisions, but the starting point might be considerably disadvantageous. As we will see later, backward elimination will also render infeasible for high-dimensional input spaces. An alternative approach to overcome this severe drawbacks is the forward selection strategy. Its theoretical aspects will be addressed in the next section. This includes also a proof of the monotone convergence property of the forward selection in combination with the general mutual information.

6.1.2 Proof of the Monotone Convergence Property of Forward Selection

As mentioned before, an alternative and feasible approach for the backward elimination algorithm, is the forward selection strategy. In forward selection, rather than starting with the full feature set and eliminating irrelevant features, relevant features are added to an initially empty set. In this case, the GMI is used to maximize the information gain instead of minimizing the information loss.

In other words, the goal of forward selection is to select a feature set $\xi_M = \{\xi_{r_1}, ..., \xi_{r_M}\}$ from a superset of N features, that provides the maximum information available for the mapping between input space ξ_M and the according output space η. A detailed description of the forward selection strategy can be found in Section 6.2.3.

In order to justify the formulation of this selection strategy as greedy algorithm and prove its convergence, its monotone convergence property will be shown in the following:

Let $I_2^{(m)} = I_2(\{\xi_{r_1}, ..., \xi_{r_m}\}, \eta)$, $m = 1, ..., M$, $M \leq N$ be the value of the mutual information in the m-th iteration of the forward selection process. Since $I_2^{(m)}$ cannot exceed the value of the global optimum $I_2^{(opt)}$, the sequence of mutual information values is bounded above and $I_2^{(m)} \leq I_2^{(opt)}$ holds for all $m = 1, ..., M$. The global optimum $I_2^{(opt)}$ is obtained by per-

forming an exhaustive search over all possible feature combinations, as described in Section 6.2.1.

Without loss of generality, there exists a suboptimal upper bound $I_2^{(conv)} \leq I_2^{(opt)}$ for the sequence $I_2^{(m)}$. Let us remember that any monotonically increasing sequence with an existing upper bound is convergent. Since the sequence of mutual information values has a defined upper bound, its convergence can be proved by showing the monotony $I_2^{(0)} \leq I_2^{(1)} \leq ... \leq I_2^{(m)} \leq ... \leq I_2^{(M)}$, $M \leq N$ of the sequence values.

With the above definitions, the following holds:

$$\frac{I_2^{(m+1)}}{I_2^{(m)}} = \frac{I_2^{(m)} + G^{(m)}}{I_2^{(m)}} = 1 + \frac{G^{(m)}}{I_2^{(m)}} \geq 1, \tag{6.3}$$

where

$$\begin{aligned} G^{(m)} &= \max_{k=1,...,N} \left(I_2(\{\xi_{r_1}, ..., \xi_{r_m}\} \cup \{\xi_k\}, \eta) - I_2(\{\xi_{r_1}, ..., \xi_{r_m}\}, \eta) \right) \\ &= \begin{cases} 0 & : \quad I_2(\{\xi_{r_1}, ..., \xi_{r_m}\} \cup \{\xi_k\}, \eta) \leq I_2(\{\xi_{r_1}, ..., \xi_{r_m}) \\ > 0 & : \quad else. \end{cases} \end{aligned} \tag{6.4}$$

The information gain $G^{(m)}$ in the m-th iteration of the selection process cannot be negative, since for $k = 1, ..., N$, it always exists at least one variable $\xi_k \in \{\xi_{r_1}, ..., \xi_{r_m}\}$, which has already been selected. Since this variable is in the set of previously selected features, no further information is obtained with this variable and the information gain $G^{(m)}$ becomes zero. In all other cases, the maximum achievable information is greater than zero. This fact implies the monotony of the sequence of mutual information values. In conjunction with the existence of an upper bound, the convergence of the sequence $I_2^{(m)}$ can be inferred and the following holds:

$$\lim_{n \to N} I_2^{(n)} = I_2^{(conv)}. \qquad \square \tag{6.5}$$

The monotony of the sequence of mutual information values justifies also the formulation of the forward selection strategy as greedy algorithm. Since

the mutual information can only increase from iteration to iteration, a locally made decision does not have to be reconsidered again in later iteration steps. Either more information is obtained or the iteration process converges.
However, to what extend the suboptimal solution $I_2^{(conv)}$ reaches the global optimum $I_2^{(opt)}$ is a different matter. In the next section, it will be explicitly demonstrated how the forward selection strategy in combination with the GMI provides a clean and straightforward solution for real-world feature selection problems. It will be shown, that the obtained results from this selection strategy are very close to the global optimum solution.
Since this selection strategy also maintains computability on a large scale, there is a clear trade-off between the rather small disadvantage of a suboptimal solution and the computational expenses. This fact clearly favors the forward selection strategy versus the backward elimination algorithm.

6.2 Assessing Feature Selection Strategies with the GMI

It has already been mentioned in Section 6.1, that the choice of a feature selection strategy is assumed to have a considerable influence on the quality of the obtained outcomes, i.e. the information content of the selected features. To reveal the extent of this influence on the basis of real measuring data, the backward elimination- and the forward selection strategy are both applied to industrial glass melting process data.

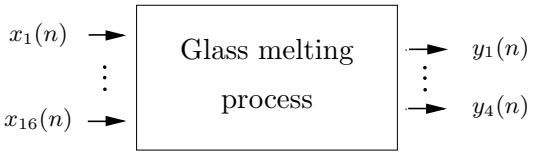


Figure 6.1: *System model industrial glass melting process.*

With respect to the information content measured by the GMI, the results of both selection strategies are finally compared to the global optimal solution.

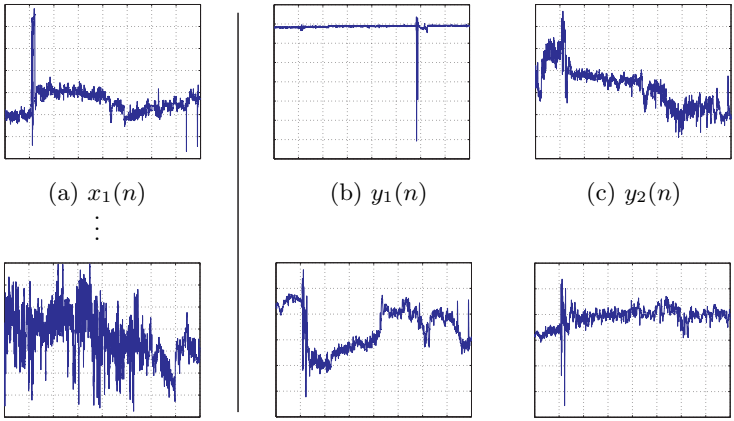
Figure 6.1 depicts an abstract system model of the previously mentioned industrial glass melting process. The variables x_i are the inputs of the open-loop process control. The output variables y_j represent the response from the system due to the applied inputs. As a matter of course, the glass production process itself is kept a company secret of Schott Glass AG. Hence, the available measuring data[1] has been scaled and biased to prevent its technical interpretability. Since the GMI method does not necessarily dependent on the interpretability of data as physical quantities, this does not imply any restriction to the applicability of the GMI method in this case.

However, the focus of this section is set on the various strategies to identify the features, i.e. a subset of the input variables, that have significantly high information content with respect to the observable system response. During the feature selection process, irrelevant or redundant features are removed and the selected variables are ordered due to their information content. The resulting feature set is then employed to construct a training data set for further neural process identification.

Figure 6.2 depicts the qualitative characteristics of the glass melting process data. The data has already been preprocessed to remove outliers and to fill up data gaps due to failures in the employed measuring equipment. Finally, Savitzky-Golay Filtering has been applied to smooth out points of discontinuity generated by the insertion of missing values. The data set contains 16 input- and 4 output signals, while each signal is comprised of 16000 data points. The sampling interval between the data points is 30 minutes.

The following sections are dedicated to the description of various ways of feature selection on the basis of the general mutual information, which has been extensively described in Chapter 5. The investigations of the various feature selection strategies are performed with the data set depicted in Figure 6.2.

[1] The glass melting process data has been made available to the public by Schott Glass AG in the context of the "EUNITE 2004" competition.



(a) $x_1(n)$ (b) $y_1(n)$ (c) $y_2(n)$

(d) $x_{16}(n)$ (e) $y_3(n)$ (f) $y_4(n)$

Figure 6.2: *Qualitative characteristics of the preprocessed glass melting data. (a), (d) First and last of the 16 inputs to the glass melting process. (b), (c), (e), (f) Response from the system due to the applied inputs.*

6.2.1 Global Selection Strategy

As previously mentioned, various feature selection strategies are feasible while performing data analysis with the GMI method. In this section, the global selection strategy is combined with the GMI and applied to the glass melting process data depicted in Figure 6.2.
Since the global selection strategy is a so called *brute force* method, it does always find the global optimum, with respect to the combination of input variables with maximum GMI value among all other possible constellations. However, the price to achieve this optimal solution has to be paid by means of tremendous computational expenses.

Suppose a feature subset containing D variables has to be selected form a superset of N input variables, then the number of distinct combinations of input variables is

$$G(N, D) = \sum_{k=1}^{D} \binom{N}{k}. \tag{6.6}$$

In the case of our glass melting process data, the realizations of the input variables imply time dependence with respect to the process output. To account for such time dependencies, each input variable might be associated with an individual time lag before GMI analysis is applied. Since every constellation of variables has to be combined with all possible time lag constellations, the total number of distinct combinations in this case extends to

$$G(N, D, T) = \sum_{k=1}^{D} \binom{N}{k} (T+1)^k, \tag{6.7}$$

where N refers to the number of available input variables from which the D dimensional subset has to be selected. The maximum number of considered time lags for all variables is expressed by T.

It can be seen very easily in Table 6.1, that even for considerably small values for N, D and particularly for T, the number of combinations dramatically increases. Table 6.1 depicts the rapidly increasing combinatorial

Feature Set Dimension N	Feature Subset Dimension D	Time Lags T	GMI Values G	estimated Flops $\lfloor F \rfloor$
16	1	0	16	$16.80 \cdot 10^9$
16	2	0	136	$142.80 \cdot 10^9$
⋮	⋮	⋮	⋮	⋮
16	**7**	**0**	**26332**	$\mathbf{27.65 \cdot 10^{12}}$
⋮	⋮	⋮	⋮	⋮
16	16	0	65535	$68.81 \cdot 10^{12}$
16	1	1	32	$33.60 \cdot 10^9$
⋮	⋮	⋮	⋮	⋮
16	**7**	**1**	**2150720**	$\mathbf{2.26 \cdot 10^{15}}$
⋮	⋮	⋮	⋮	⋮
16	**7**	**48**	$\mathbf{7.87 \cdot 10^{15}}$	$\mathbf{8.26 \cdot 10^{24}}$

Table 6.1: *Combinatorial complexity of the global selection strategy as a function of the dimension of the feature set-, the feature subset and the maximum number of time lags.*

complexity by means of the number of input variable combinations. The number of variable combination, and thus the number of required GMI values, is a function of the feature set dimension N, the feature subset dimension D and the maximum number of considered time lags T. To give an impression for the remarkable computational complexity, a lower limit F for an estimate of the required number of floating point operations is additionally listed in the last column of the table.
The emphasized lines in Table 6.1 show situations of particular interest for the applicability of the global selection strategy. The situation where $N = 16,\ D = 7$ and $T = 0$ turned out to be the limit of practical computability for the global selection strategy on currently available standard personal computers. If time dependencies between inputs and outputs also have to be considered in the analysis of the process data, the computational complexity dramatically increases.
For the similar parameter constellation $N = 16,\ D = 7$ but this time with $T = 1$, the computational complexity with respect to the required GMI calculations rises from 26332 to 2150720. In case of the available glass melting process data, the consideration of at most one time lag is not really useful for the identification of potential time dependencies in the data set as a whole. A more realistic setting for the identification of time dependencies would be $N = 16,\ D = 7$ and $T = 48$. On behalf of the available process data, this would allow the analysis of time dependencies ranging back up to 24 hours. However, this implies instantly the necessity for the computation of $7.87 \cdot 10^{15}$ GMI values, which is by far beyond any practical computability.

In any case, as a consequence of this immense complexity, the identification of time dependencies renders impossible while employing the global selection strategy. Instead of that, the optimum feature subsets for the settings $N = 16,\ D = 1, ..., 7$ and $T = 0$ are calculated for each output variable.

	x_{i_1}	x_{i_2}	x_{i_3}	x_{i_4}	x_{i_5}	x_{i_6}	x_{i_7}	$I_2(x, y_1)$
1	1	0	0	0	0	0	0	0.1342
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
23259	3	4	6	7	9	10	11	0.6138
23260	3	4	6	7	9	10	12	0.6308
23261	**3**	**4**	**6**	**7**	**9**	**10**	**13**	**0.7333**
23262	3	4	6	7	9	10	14	0.6547
23263	3	4	6	7	9	10	15	0.6493
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
26332	10	11	12	13	14	15	16	0.3724

Table 6.2: *Indices of the input variable constellation around the maximum GMI value $I_2(x_{glob}, y_1)$ among all possible input combinations.*

In Table 6.2, the indices of the input variable constellation and their resulting GMI value $I_2(x, y_1)$ are listed. The set of input variables x in the expression $I_2(x, y_1)$ is defined as $x = \{x_{i_k} | i_k \neq 0,\ k = 1, ..., D\}$, where D is the cardinality of the feature subset, which contains the chosen variables from the N dimensional superset of all features. In this particular case, the number of variables to be chosen is restricted to $D = 7$, due to the previously mentioned computational limitations. Finally it turned out, that for the output y_1 the input variable combination x_3, x_4, x_6, x_7, x_9, x_{10}, x_{13} has the highest GMI value among all other combinations consisting of up to seven variables.

In the context of optimal feature selection, the selected feature subset essentially has to fulfill two requirements. On the one hand, the selected variables ideally contain the maximum amount of information. Since the input variables that maximize the GMI are chosen from all possible combinations, this demand is of course implicitly met when employing the global selection strategy.
On the other hand the set of selected variables has to be as small as possible, but also as large as necessary to solve the underlying problem. If too few variables are selected, important information might be unnecessarily ignored. On the opposite, any additional or redundant variable might obscure the information of previously selected features.

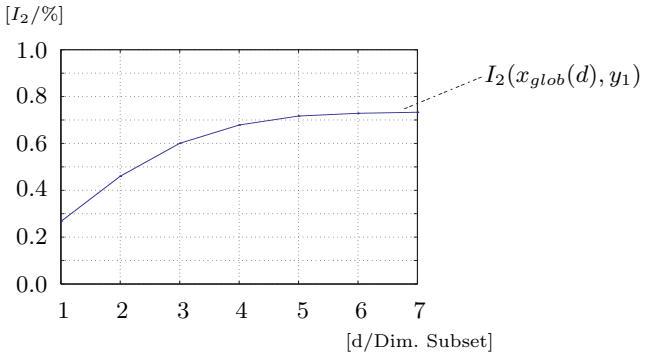


Figure 6.3: *The maximum GMI $I_2(x_{glob}(d), y_1)$ as a function of the maximum number of selected variables in the feature subset.*

In Figure 6.3, the maximum GMI values $I_2(x_{glob}(d), y_1)$ are depicted as a function of the cardinality d of the selected feature subset.
The set of input variables $x_{glob}(d)$ maximizes the expression $I_2(x_{glob}(d), y_1)$ dependent on d. In conjunction with Table 6.2 it is defined as

$$x_{glob}(d) = \underset{v_i}{\operatorname{argmax}} \left[I_2(v_i, y_1) \right], \tag{6.8}$$

where $v_i = \{x_{i_k} \mid i_k \neq 0,\ k = 1, ..., d\}$, $i = 1, ..., G(N, d)$.
It can be clearly observed that, the maximum GMI values at least tend asymptotically to an upper bound, which is specific to the underlying data. This fact obviously implies that at a certain point, no further information gain can be achieved when adding more variables to the subset of features. This means that, the most significant variables have already been selected at this point. In the example above, this point is reached at $d = 7$. It defines the tradeoff between the number of input variables and the highest obtainable information content. Hence, it also defines the optimal number of input variables where no information is omitted and no variables are unnecessarily enclosed. This specific example shows that the upper bound of the GMI is already reached when employing the seven input variables particularly emphasized in Table 6.2.
In Appendix C.1, the computation results for all output variables y_1, y_2, y_3 and y_4 can be found. Figure C.1 also summarizes the curves of the maximum GMI values for all output variables.

Finally, it can be concluded that the global selection strategy does definitely find the unique optimal combination of input variables with respect to the information content measured by the GMI. However, this goal is attained at the price of disproportional high computational expenses. Due to the immense combinatorial complexity, the identification of potential time dependencies within the available glass melting data set turns out to be infeasible. As another consequence of the combinatorial complexity, the global selection strategy is not applicable to feature selection from large sets of input variables, such as shown in Section 7.2.3.

In the following, the backward elimination- and the forward selection strategy, both with considerable reduced computational requirements, are investigated. Since both strategies are assumed to provide suboptimal solutions, their outcomes are compared to their corresponding results of the global selection strategy. It will turn out that the assumedly suboptimal solutions, in particular from forward selection, are absolutely comparable to the optimal results from the global selection strategy.

6.2.2 Backward Elimination Strategy

Different from global selection, the backward elimination strategy does not find the combination of variables that implies the global optimum with respect to the GMI. In fact, it produces a suboptimal solution, which is stated by KOLLER and SAHAMI [KS96] to be relatively close to the global optimum.

If we assume N input variables, the first iteration requires $(1 + N)$ calculations of the GMI, one with the full variable set and N calculations, each with one temporarily removed variable. Since the variable with the smallest information gain has already been removed in the preceding step, the second iteration requires just $(N - 1)$ GMI calculations and so forth. Because the selection of the last variable is trivial, the last GMI calculation might be omitted.

Hence, the number of required GMI calculations for the backward elimination strategy is

$$G(N) \;=\; 1 + \sum_{k=1}^{N} k, \tag{6.9}$$

where N refers to the number of available input variables. Compared to the global selection strategy, the number of required GMI calculations is significantly reduced.

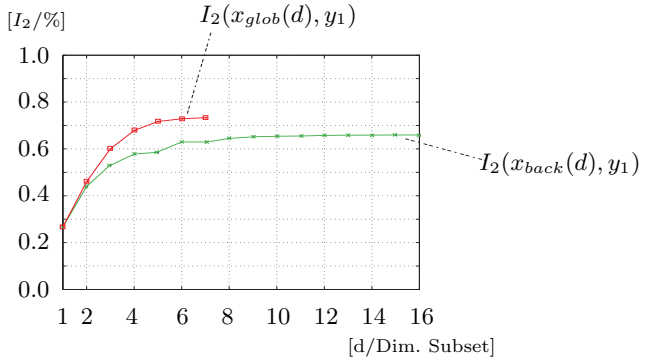


Figure 6.4: *The GMI of the chosen variables from backward elimination $I_2(x_{back}(d), y_1)$ and the global maximum $I_2(x_{glob}(d), y_1)$ as a function of the number of selected variables in the feature subsets.*

In Figure 6.4, the GMI values of the chosen variables from backward elimination $I_2(x_{back}(d), y_1)$ are compared to the maximum GMI $I_2(x_{glob}(d), y_1)$. Both GMI curves are functions of the number of selected variables d in the feature subsets and are computed on the basis of the data set depicted in Figure 6.2. Due to the previously mentioned combinatorial complexity, the curve for the global selection outcome covers only seven dimensions. It can be observed that both selection strategies show the same quantitative curve progression. Expectedly, the features that had been identified with the backward elimination strategy show less information content that those from global selection.
In Appendix C.2, the results for all output variables are depicted. However, some of the results are very close to the global optimum while others show a rather large deviation from the global optimum. This leads to the conclusion, that the quality of the results of backward elimination is rather unstable and might be highly dependent on the character of the underlying data. From the practical viewpoint of optimal feature selection, this disadvantage is undesirable.

Besides that, another drawback of the backward elimination strategy is, that it has to start out with the maximum dimension and run through all iterations before the final result can be obtained. In particular, this could cause considerable restrictions when analyzing high dimensional input data, such as demonstrated in Section. 7.2.3. In order to avoid such imponderability, the forward selection strategy has to be considered as a feasible alternative approach.

6.2.3 Forward Selection Strategy

In the case of our glass melting process data, feature selection is terminated if a previously defined threshold for the total information content is reached. In Figure 6.5, the GMI values of the chosen variables from

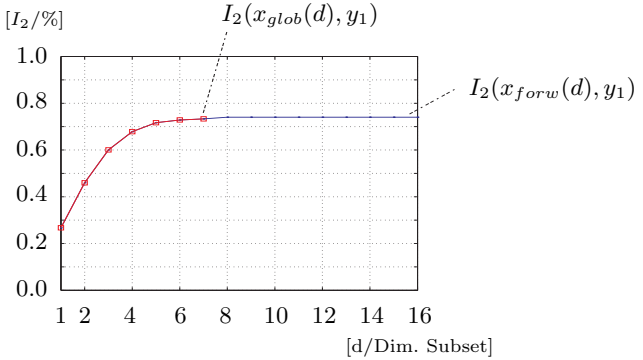


Figure 6.5: *The GMI of the chosen variables from forward selection* $I_2(x_{forw}(d), y_1)$ *and the global maximum* $I_2(x_{glob}(d), y_1)$ *are depicted as a function of the number of selected variables in the feature subsets.*

forward selection $I_2(x_{forw}(d), y_1)$ are compared to the maximum GMI $I_2(x_{glob}(d), y_1)$. In this case, it can be clearly observed that, the result of forward selection is identical to the global optimum solution obtained through the global selection strategy. In Appendix C.3, the results for all output variables y_1, y_2, y_3 and y_4 are compared to the optimum solution. Figure C.7, depicts a slight deviation of the forward selection solution from the global optimum. However, in contrast to the backward elimination strategy, forward selection comes up with reliable good solutions, which are remarkably close to the global optimum. The constant high quality of the forward selection strategy can be clearly seen by comparing the results for output variables y_1 and y_4 with its equivalents from backward elimination.

The high quality of the results and the significantly reduced combinatorial complexity of the forward selection strategy, facilitates the previously mentioned identification of process immanent time dependencies.
Since forward selection can be also terminated if a predefined number D of

features has been selected from the set of available variables, the number of required GMI calculations is

$$G(N, D, T) = D \cdot N \cdot (T + 1), \tag{6.10}$$

where $D \leq N$ is again the dimension of the feature subset, N refers to the number of available input variables and T denotes the maximum number of time lags to be considered.
Since the forward selection strategy starts out with an empty feature set and accounts for the information gain obtained by adding new features, there is no need to run through the complete iteration process like in backward elimination. The iteration process could be terminated after the predefined number of features has been selected. Furthermore, as a consequence of the reduced computational expenses, the forward selection strategy provides also the possibility to analyze the occurrence of time dependencies in the underlying data.

This section revealed, that forward selection is a feasible and robust alternative to the global- and backward elimination strategy, while maintaining computability on a large scale. It turned out that the results from forward selection are constantly close to the global achievable optimum solution. This renders forward selection feasible for the analysis of high dimensional data such as accomplished in Section 7.2.
In Section 6.3, the forward selection strategy will be employed for the analysis of our glass melting process data, which has already been introduced at the beginning of this chapter. The input variables, which have been selected based on an established information theory concept, will be further used for the construction of neural training set. Finally, it will be demonstrated in Section 7.1, how these training sets can be applied to perform a neural identification of the glass melting process behavior.

6.3 Feature Selection employing Forward Selection and the GMI

For different selection strategies, their performance and quality issues have been investigated in Section 6.2. It turned out, that the forward selection strategy provided the best results with respect to quality, while maintaining computability on a large scale [HF01]. In the following, this selection strategy will be employed to accomplish optimal feature selection from the previously introduced glass melting process data. In Chapter 7, it will be demonstrated how the GMI based feature selection can easily be employed for further types of technical problems.

6.3.1 Forward Selection without the consideration of Time Lags

As mentioned earlier in this chapter, the forward selection strategy can be applied in multiple ways for feature selection. In Section 6.2.3, the forward selection strategy has been described in a rather formal way. In order to demonstrate how this strategy can be brought to life, feature selection is initially performed without the intention of identifying time dependencies. In a second step, the time-dependent aspect will also considered in the analysis of the process data.

When considering the forward selection procedure as described in Equation 3.40, it is quite clear that the selection of variables is performed by passing through multiple iteration loops. Figure 6.6 depicts the situation at the beginning of the first iteration for output variable y_1. At this time, the subset of selected features is still empty. Since the goal is to determine the variable providing the maximum information gain in each iteration, the GMI values $I_2(\{x_i(n)\}, y_1(n))$ are calculated for $i = 1, ..., 16$, i.e. for all available input variables. After all values have been calculated, the input variable with the highest GMI is selected. This variable will then be used as a fixed input for all GMI calculations in the succeeding iterations. Figure 6.7 illustrates the outcome of the first iteration cycle with respect to the general mutual information. The so called *GMI function* plots the GMI values against the variables x_i, which have been consecutively added to the initially empty feature subset. In this case, the variable x_7 provides the highest GMI value and is therefore the first variable to be selected.

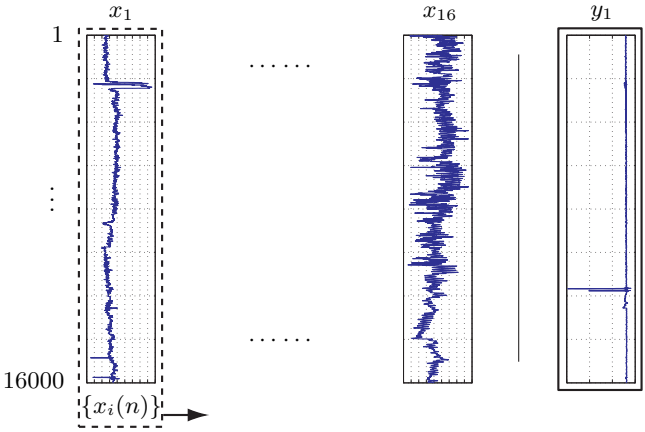


Figure 6.6: *The first iteration of the feature selection procedure: Calculation of the GMI between the one dimensional time series* $\{x_i(n)\}$ *and the time series of the output variable* $y_1(n)$.

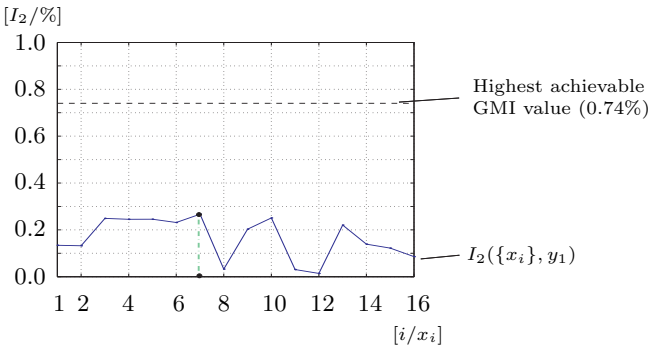


Figure 6.7: *The GMI function of the first iteration cycle for output variable* y_1.

The dashed line in Figure 6.7 marks the maximum achievable GMI value when employing the forward selection strategy, with respect to the first output variable y_1.

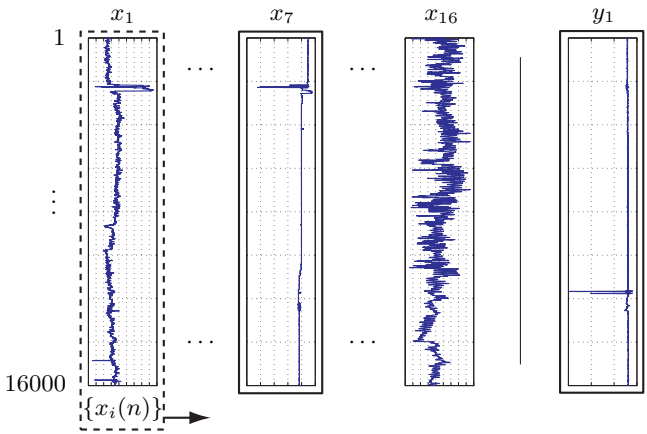


Figure 6.8: *The second iteration of the feature selection procedure: Calculation of the GMI between the two dimensional time series* $\{x_i(n), \{x_7(n)\}\}$ *and the time series of the output variable* $y_1(n)$ *of the glass melting process.*

Figure 6.8 depicts the situation at the beginning of the second iteration. From now on, the previously selected variable x_7 is a permanent input to all GMI calculations. The GMI is now calculated between the two dimensional time series $\{x_i(n), \{x_7(n)\}\}$ and the time series of the output variable $y_1(n)$. After the GMI values have been calculated for $i = 1, ..., 16$, the input variable with the highest GMI is again selected for the next iteration and so forth. Thus, the number of selected features constantly increases from iteration to iteration.

The result of the first six iteration cycles is depicted in Figure 6.9. Each iteration produces a so called GMI function, which is used to determine the input variable to be currently selected. As mentioned before, the iteration process might be terminated if a predefined GMI threshold is reached. In the case of our glass melting data, this threshold is defined to be 99% of the maximum achievable GMI value for the current output variable.
However, one might ask for what reason this threshold has to be chosen at all. In certain cases, the forward selection strategy tends to produce identical GMI functions, if the values approach the highest obtainable GMI value. If this happens, an additional variable does not imply further information gain and the iteration has to be stopped anyway. Otherwise the

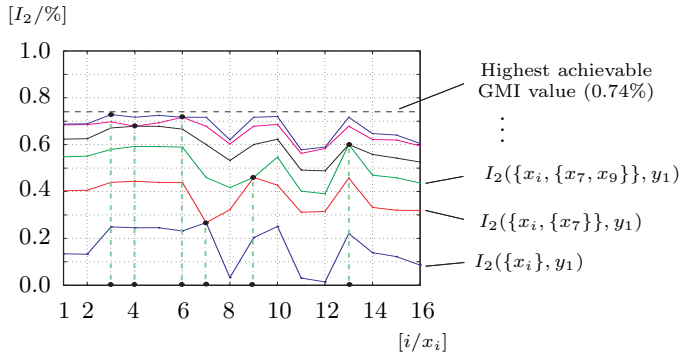


Figure 6.9: *The GMI functions of the first six iteration cycles for output variable y_1. The variables comprising 99% of the maximum achievable GMI are: x_7, x_9, x_{13}, x_4, x_6, x_3.*

same variable would be selected over and over again, which does not provide any information gain either. It could be observed that this behavior did occur most frequently, if the GMI functions of consecutive iteration cycles look very similar right at the beginning of the analysis.

In this context, another interesting fact has to be noticed. If we consider the first three iterations as depicted in Figure 6.9, it can be observed that the GMI functions $I_2(\{x_i\}, y_1)$ and $I_2(\{x_i, \{x_7\}\}, y_1)$ only share one common result, namely the GMI value $I_2(\{x_7\}, y_1)$ of the recently selected variable x_7. Without exception, all GMI values of the second iteration cycle are greater or equal to the maximum of the preceding iteration. This means that any variable, apart from x_7, will definitely come along with a positive information gain.
However, a different situation can be observed if we consider the GMI functions $I_2(\{x_i, \{x_7\}\}, y_1)$ and $I_2(\{x_i, \{x_7, x_9\}\}, y_1)$. The latter points out the variables x_8, x_{11}, x_{12} and x_{16}, which would clearly cause a loss of information if selected. As a matter of fact, this means that only specific variables will imply a benefit with respect to information gain, while others do negate already present information content.

When we recall again the outline of the backward elimination strategy, the existence of such variables would result in an incorrect starting point for

the entire selection process. Remember that the bottom line of backward elimination is the assumption of a proper starting point distribution. The designated goal in each iteration is to stay as close as possible to this starting point distribution, while removing variables. In other words, backward elimination is trying to preserve as much of the initial information as possible, while starting out with the complete variable set and assuming it to be the ideal constellation.
Although the optimality of the backward elimination strategy itself has been established in Section 6.1, the starting point of the iteration seems to be the critical factor. The prove in Section 6.1, only concludes that the backward elimination strategy yields an optimal solution. However, this does not take into account the information content of the initial variable set. Since backward elimination always has to start out with a complete set of variables, the total information content of this combination could be considerably reduced due to the degradation effect described above.
Altogether, different from the forward selection strategy, this appears to be the actual reason for the partially poor performance of backward elimination in practical applications.

6.3.2 Forward Selection including Time Lags

It had been mentioned earlier in this chapter, that the forward selection strategy can be applied in various flexible ways for data analysis. Besides different termination conditions of the iteration process, the reduced computational complexity, as denoted in Equation 6.10, also allows for the analysis of time dependencies.
In this section, it will finally be demonstrated how forward selection in combination with the GMI could be efficiently employed for the analysis of data immanent time dependencies.

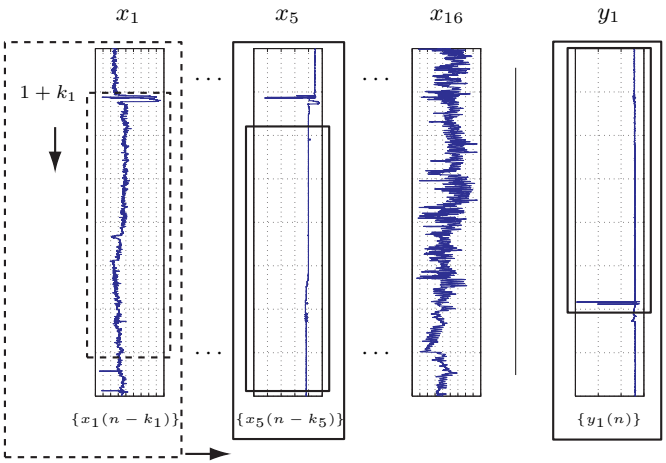


Figure 6.10: *The second iteration of the feature selection procedure including the analysis of time dependencies.*

Figure 6.10 depicts the second iteration of the feature selection process, which also includes the analysis of temporal dependencies. Compared to the procedure illustrated in Section 6.3.1, data windows are employed now to obtain variable specific time delays k_i. More precisely, if the variable x_i is currently considered, the GMI is calculated while the data window is moved over the corresponding time series with $k_i = 0, ..., T$. If we have a data set consisting of N input variables and a maximum number of T time lags should be considered, a two dimensional GMI function containing $N(T + 1)$ values is generated in each iteration. After calculation, the

variable–time lag combination that corresponds to the highest information content is selected as the fixed input for the next iteration.

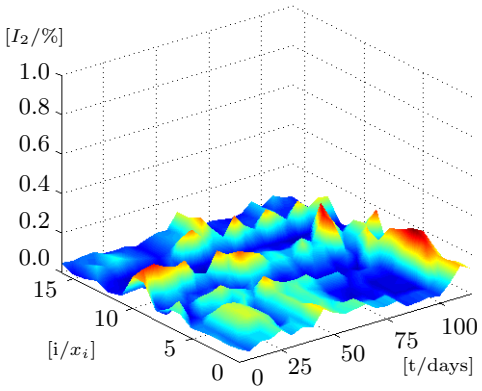


Figure 6.11: *The GMI as a function of the input variable x_i and the variable specific time lag t for output variable y_1*

Figure 6.11 depicts the GMI function of the first iteration of the forward selection strategy. The GMI values $I_2(\{x_i(n-k_i)\}, y_1(n))$ are computed as a function of the input variable x_i and a variable specific time lag k_i. In Figure 6.11, the time lags k_i are already converted to reflect days instead of arbitrary units. It can be observed that the variable x_{12} does not show any temporal dependency over the analysis interval of 110 days. This observation corresponds perfectly to the result obtained in Section 6.3.1, where the time aspect has not been considered. In fact, the variable x_{12} has never been selected as a relevant feature by any of the investigated selection strategies in this work.

It can further be perceived that the input variables x_1 through x_7 show considerable GMI values with a lag of time of approximately 110 days. As a matter of fact, variable x_5 with a time delay of 109 days has been selected as the most relevant feature in this iteration cycle.
Another interesting fact can be observed, when we draw our attention to the cutting edge of Figure 6.11, where $t = 0$. Here, it can be seen that the quantitative curve progression of the slice plane is very similar to the

according GMI function depicted in Figure 6.7. However, the difference in the absolute values of the both curves is due to the limited length of the data sequences. Under the rather unrealistic assumption of unlimited data sequences and thus under the assumption of unlimited computing resources, both GMI function would be identical.

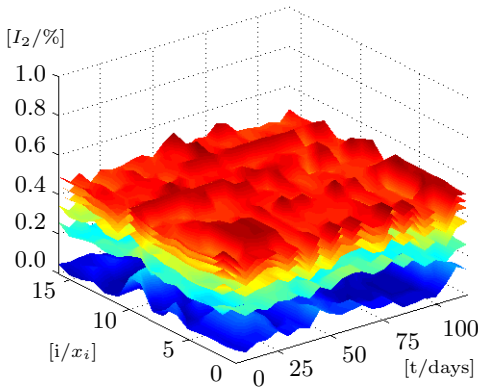


Figure 6.12: *The GMI functions of the first six iteration cycles for input variable y_1. The GMI is now a function of the input variable x_i and the time lag t for output variable y_1.*

In the end, Figure 6.12 depicts the GMI functions of the first six iteration cycles for the output variable y_1. Like already presented in Figure 6.9, it can be also observed here that the GMI functions become identical with a growing number of iteration loops. Hence, this behavior could be successively reproduced, even when additionally analyzing the temporal aspects of the glass melting process data.
In this context, it should be mentioned that the GMI function including time lags cannot be compared with the maximum achievable GMI value. The explanation for this has been presented in Table 6.1. It depicts the effect of the tremendous combinatorial complexity, when it comes to the analysis of temporal aspects. Hence, the global selection strategy could not be applied to the glass melting process data in this case.

Iteration	Variable [i/x_i]	Time Lag [t/days]	GMI Value [I_2/%]
1	x_5	**109**	0.22
2	x_{11}	99	0.34
3	x_{13}	68	0.44
4	x_7	**109**	0.49
5	x_6	15	0.52
6	x_{14}	57	0.56

Table 6.3: *The selected variables together with the according time lag for output variable* y_1.

Since the selection process tends to produce identical GMI functions similar to the results depicted in Figure 6.9, the iteration has been terminated after six loops. The selected variables and their according time lags are assembled in Table 6.3. The additional analysis of the temporal aspects revealed several long-term dependencies, which are likely to be induced by the high specific heat capacity of glass and thus the notable thermal inertia of the whole system. Hence, the results of the GMI analysis correspond perfectly to the technical reality of the glass melting process.

In this section, it could be successfully demonstrated how feature selection can be practically performed by employing the forward selection strategy for the analysis of real process data.
In the first part of this section, the analysis has been performed without the intention of identifying time dependencies within the underlying process data. In the second part, the forward selection strategy has been employed to reveal data immanent temporal correlation.
It turned out that the forward selection strategy has a wide range of applicability. The results lead to the conclusion that the forward selection strategy, in combination with the GMI, is a stable and valuable method for feature selection. In order to confirm the value of the proposed concept of feature selection, application case studies will be performed later on.

7 Application Case Studies and Proof-of-Concept

In this Chapter, two selected engineering problems are addressed with the previously introduced generalized mutual information. The first task is the analysis of the glass melting process. The second challenge is the efficient reduction of a high dimensional data space in automotive engineering. Since both tasks involve highly nonlinear behavior, the application of a nonlinear analysis method is inevitable.

7.1 Neural Identification of the Glass Melting Process

Throughout the ages, glass played an important role in the development of human society, its art- and its technological history. In the stone age, man used natural volcanic glass, also known as obsidian, as a cutting tool and to produce weapons for hunting. Artificial glass was discovered, while accidentally melting sand and natron. The oldest findings of artificially produced glass, pearls from the graves of Egyptian kings, are dated about 3500 B.C. The invention of the glass melting furnace about 200 B.C. revolutionized the manufacturing of glass and gave way to the production of flat glasses. Glass for basic ophthalmic applications was manufactured since 1250, and simple microscopes and telescopes had been produced since the 17th century.
In Benediktbeuern, the German Joseph Fraunhofer (1787-1826) sought to bring glass melting under scientific control. He improved microscopes, telescopes and survey instruments. In 1846, Carl Zeiss founded an optical shop in Jena and in 1866, he started together with Ernst Abbe to design optical instruments on a scientific basis. However, this scientific approach required high-quality glass with constant and predictable optical proper-

ties. In 1879, Dr. Otto Schott performed systematic research and identified various components and how they influenced the optical properties of technical glass. Since this time, the knowledge about glass production evolved with remarkable speed and the ancient art of glass making rapidly changed into an independent technical discipline.

In order to maintain the high quality of modern technical glass products, the rather complex production process has been focused on as a subject for investigation. In this context, the objective is to deduce a mathematical model describing the input-/output behavior of the technical glass production process from available observational data. The motivation behind this ambition is the ability to predict the properties of the glass before its production does even occur. This chapter is dedicated to the neural identification of the glass production process based on the outcome of a feature selection strategy described in Section 6.3.

7.1.1 Industrial Glass Production

The physical definition of glass is a solid, uniform and amorphous material. It is usually obtained when a suitably viscous molten material cools down very rapidly, thus no regular crystal lattice could form. In its natural form, glass is a transparent, relatively strong, hard-wearing and biological inactive material. However, untreated glass is brittle and will break into sharp shards. These properties can be modified, or even changed entirely, with special additives or adequate heat treatment.

Common glass is mostly amorphous silicon dioxide (SiO_2), which is basically the same chemical compound as found in natural quartz. The melting point of pure silica is approximately 1800°C. In order to reduce the melting point to about 1000°C, soda (Na_2CO_3) or potash (K_2CO_3) is added. However, soda makes the glass water-soluble. Since this is undesirable, calcium oxide (CaO) is added to restore its insolubility. In Figure 7.1, the interior of a glass melting tank is depicted. The picture was taken at Schott Glas AG in Mainz, Germany and shows technicians accomplishing the periodic maintenance of the tank after cooling down.

Figure 7.2(a) depicts the flow of the liquid glass in a glass melting tank. The raw material enters the melting tank through the batch and melts as it flows towards the center. It can be seen, that in the middle of the glass tank, air is injected. This so called *bubbling*, yields a forced convection which removes the parasitic inclusions from the melted mass. After pass-

Figure 7.1: *Glass melting tank after cooling down.*

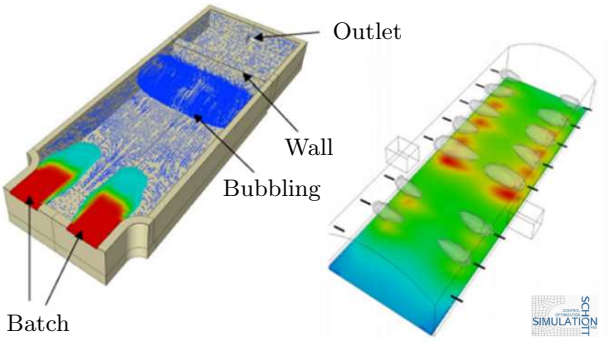


Figure 7.2: *Technical specification of a glass melting tank. (a) Glass melting with embedded flow vectors. (b) Combustion chamber with temperature distribution.*

ing a wall which serves as a siphon the mass is ready for further processing and leaves the tank through an outlet.

At both sides of the combustion chamber, the blowers are located which provide the necessary heat for the melting process. Mostly a combination of heavy oil and pure oxygen is used as energy source for the heating system. Figure 7.2(b) gives an impression of the temperature distribution on the surface of the molten glass. It seems noteworthy that the temperature over the convection zone is significantly higher than in the rest of the combustion chamber. Since the viscosity of glass is inversely proportional to the temperature, parasitic air inclusions could leave the glass more easily at this part of the surface.

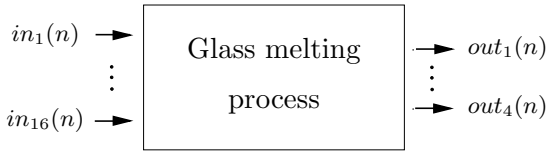


Figure 7.3: *System model of the glass melting process.*

In Figure 7.3, an abstract system model of the previously described glass melting process is outlined. The variables in_j are the input signals for the open-loop control of the glass melting process. The variables out_k represent the system response and can thus be used as direct quality measures of the entire melting process. An overview of the glass melting process data is depicted in Figure 6.2. Since the technical details of the process are strictly kept a company secret, the raw data set containing the input- and output signals has been scaled and biased to conceal its technical interpretability.

7.1.2 Neural Process Identification

In this section, the outcomes of the forward selection strategy are employed to accomplish a neural identification of the glass melting process. However, when additionally considering time lags, only approximately 30% of the already rare measurement data could be used for the construction of the neural training sets. Prior feasibility studies showed that such a small amount of training patterns does entirely yield unsatisfactory neural identification results.

As a consequence of the limited length of the available measurement data, the consideration of time lags renders infeasible for the construction of neural training sets and thus for subsequent training. Hence, the neural identification has to be performed with the results obtained in Section 6.3.1, that is without the consideration of time lags.

7.1.2.1 Neural Training Setup

After the selection of the relevant input variables has been completed, the neural data sets are specifically constructed for each training run. For this, the preprocessed glass melting process data, as depicted in Figure 6.2, is divided into separate training- and test data sets.
However, the usage of a validation data set, i.e. the presentation of unknown data during training, had been renounced for two reasons. First of all, the previously mentioned limited data availability does not allow a further reduction of the actual training data set. Secondly, we are not primarily interested in the evaluation or the performance aspects of some specific neural training algorithm.

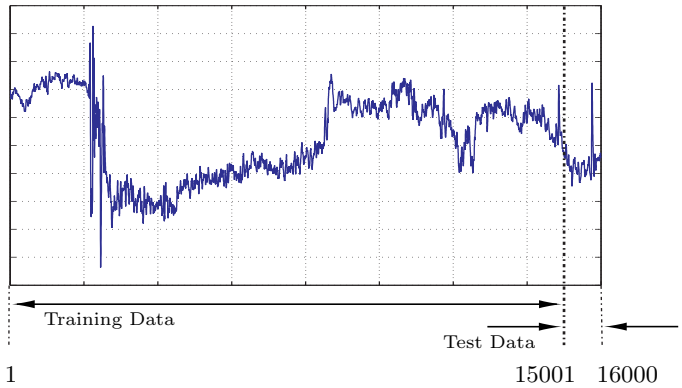


Figure 7.4: *Exemplary segmentation of the available process data into training- and test data sets.*

Hence, a separate validation data set would not generate any benefit with respect to the main focus, which definitely lies on the proof-of-concept of the proposed feature selection method based on the GMI.
Figure 7.4 exemplarily illustrates the segmentation of the available process data into training- and test data sets. It can be observed that the training set is comprised of the first 15000 patterns, whereas the last 1000 are used as test data.

In order to perceive the effect of prior feature selection on the identification results, the neural training is initially performed with all available input variables. In a second step, the GMI selected input variables are employed for training and the noticeable differences are discussed below.

Network Topology	Input Variables	Output Variables	Learning Algorithm
FF-16-16-4	$x_1, ..., x_{16}$	y_1	EKF
FF-16-16-4	$x_1, ..., x_{16}$	y_2	EKF
FF-16-16-4	$x_1, ..., x_{16}$	y_3	EKF
FF-16-16-4	$x_1, ..., x_{16}$	y_4	EKF

Table 7.1: *Neural training setup for all available input variables.*

Network Topology	Input Variables	Output Variables	Learning Algorithm
FF-6-16-4	x_7, x_9, x_{13}, x_4, x_6, x_3	y_1	EKF
FF-8-16-4	x_{10}, x_6, x_7, x_{11} x_{12}, x_{14}, x_8, x_{16}	y_2	EKF
FF-7-16-4	x_{13}, x_7, x_9, x_2 x_8, x_4, x_{16}	y_3	EKF
FF-5-16-4	x_6, x_9, x_7, x_3 x_{13}	y_4	EKF

Table 7.2: *Neural training setup for the GMI selected input variables.*

In Tables 7.1 and 7.2, the employed network topologies, the input- and output variables and the learning algorithms are listed. In all cases, the learning rate has been set to 0.05 and the neural training was conducted with the Extended Kalman Filter over 100 epochs.

7.1.2.2 Identification Results

The neural identification results, with respect to the cases in Table 7.1, are depicted in Figure 7.5. Since the measurement data has been biased and scaled to prevent its technical interpretability, the output variables are quantified with so called *arbitrary units (a.u.)*.

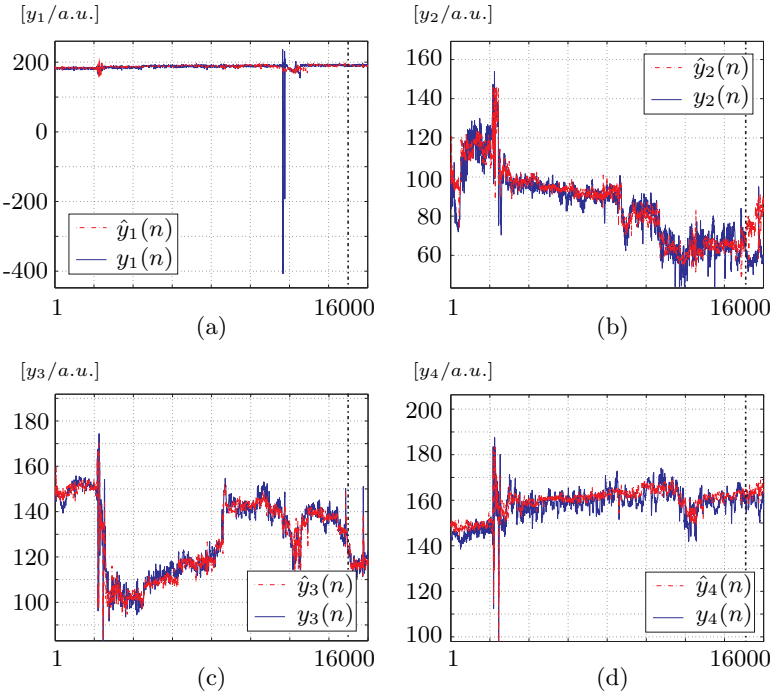


Figure 7.5: *Neural identification results with all available input variables: (a)* $|\mu_{test}|$*=1.9,* σ_{test}*=2.2, (b)* $|\mu_{test}|$*=16.3,* σ_{test}*=9.5, (c)* $|\mu_{test}|$*=0.4,* σ_{test}*=4.5, (d)* $|\mu_{test}|$*=4.2,* σ_{test}*=3.9*

After convergence, the 15000 training patterns are presented to the neural network along with the 1000 unknown test patterns. Assuming the test data to be some hypothetical input sequence to the process, the response

of the neural network could be interpreted as a prediction of the glass melting process. The quality of identification is measured by means of the absolute value of the average approximation error $|\mu_{test}|$ and its according standard deviation σ_{test}. As mentioned before, the test data ranges from pattern 15001 to pattern 16000.

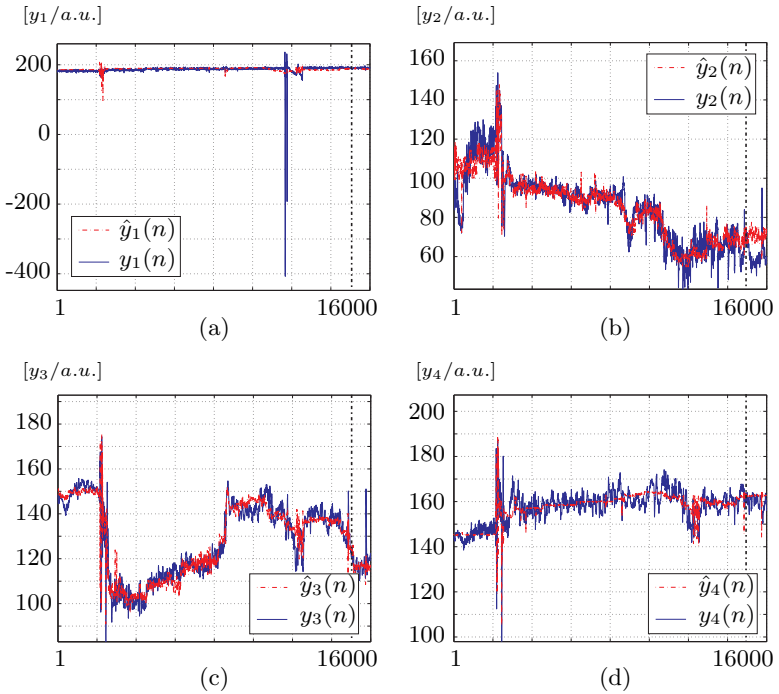


Figure 7.6: *Neural identification results with the GMI input variables: (a) $|\mu_{test}|$=1.1, σ_{test}=2.2, (b) $|\mu_{test}|$=10.6, σ_{test}=7.1, (c) $|\mu_{test}|$=0.4, σ_{test}=4.1, (d) $|\mu_{test}|$=2.1, σ_{test}=3.2*

Figure 7.6 shows the results of the identification with the GMI selected input variables. It can be observed, that for all variables the average approximation error $|\mu_{test}|$ and its standard deviation σ_{test} decreased, when employing the GMI selected input variables.

The worst results, with respect to $|\mu_{test}|$ and σ_{test}, has been obtained for variable y_2 followed by y_4. However, the best results can be observed for outputs y_1 and y_3.

In this context, one might finally also raise the question about the performance of the final system identification based on GMI feature selection, in contrast to the results obtained during the EUNITE competition. The following error function has been employed to evaluate and compare the performance of the various EUNITE solutions:

$$E_{eunite} = \frac{1}{4}\sum_{i=1}^{4}\frac{100}{N}\sum_{n=1}^{N}\frac{\|y_i(n)-\hat{y}_i(n)\|}{\|y_i(n)\|}\, f(n), \tag{7.1}$$

where

$y_i(n)$	is the desired output of variable i at time step n,
$\hat{y}_i(n)$	is the estimated output of variable i at time step n,
$N = 1000$	is the length of the error integral and
$f = \frac{500}{500+n}$	is an error weight function decreasing with the number of time steps n.

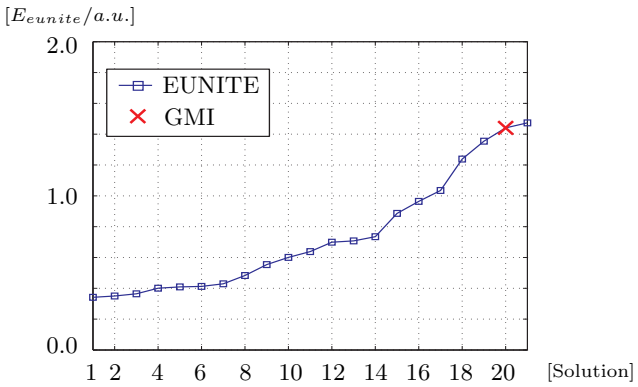


Figure 7.7: *Overall errors of the best EUNITE solutions and of the result obtained from the GMI selected input variables.*

The error function in Equation 7.1 is directly applied to the test data ranging from pattern 15001 to 16000. For the situation depicted in Figure 7.5,

this yields the overall error value $E_{eunite} = 2.87$. This measures the quality of the process prediction of 1000 time steps on the basis of all available input variables. However, in the case of GMI selected input variables, the prediction error $E_{eunite} = 1.44$ is obtained. This clearly demonstrates the improvement in modelling by the employment of the GMI feature selection. In Figure 7.7, the overall prediction error of the solution obtained from the GMI selected input variables is compared to the prediction errors of the best EUNITE solutions. It can be observed that the solution obtained from the GMI selected input variables is just among the best 20 solutions. It turned out that the quality of the obtained system model could be considerably enhanced through the application of the prior GMI feature selection process.

In order to demonstrate the wide applicability of this method, it is further assigned for a fairly complex problem in automotive engineering. The following section is concerned with the optimization of the so called neural combustion control for spark ignition engines, using the GMI driven feature selection method. In a separate section, the quality of the neural combustion model, which has been derived from the GMI selected features, is compared to its PCA- and ICA pendents.

7.2 Neural Combustion Control in Automotive Environments

7.2.1 Fundamentals of Motormanagement

In the second half of the 19th century, Nikolaus August Otto presented the first gas powered four-stroke cycle engine at the 1878 World Fair in Paris. This spark-ignition engine, which was later named after its inventor, provided the possibility of converting chemically stored energy into kinetic energy through internal combustion.
Sixteen years later, Rudolf Diesel patented the first internal combustion engine that uses oil for fuel. It was later also named after its inventor. The basic Diesel engine does not use a carburetor or an ignition system, i.e. spark plugs, but injects diesel oil directly into the cylinders when the piston has compressed the air so tightly that it is hot enough to ignite the diesel fuel without a spark. Because a cold engine cannot ignite the diesel fuel, glow plugs are used to heat the mixture.
Both revolutionary concepts differ in the way the combustion process is initiated. In the case of Diesel engines, fuel is directly injected into the combustion chamber which already contains compressed fresh air. In spark-ignition engines, a fuel-air mixture is pulled into the combustion chamber. After compression, the mixtures is ignited with an electrically generated spark.
Since the investigations and results presented in this work are exclusively related to the spark-ignition engine, this chapter intents to provide a short survey of its technical aspects. The technical information on the functional structure and management of spark-ignition engines are mostly taken from [Bos94].

7.2.1.1 Spark-ignition Engine

For more than hundred years after Karl Benz first employed a combustion engine for the propulsion of a vehicle, the two principles introduced above are still prior art for the use in passenger cars, ships and aircrafts. While a modern combustion engine cannot be compared to its beginnings, the basic principle is still the same.
Figure 7.8 depicts the basic structure of a four-stroke cycle engine. In the next section, the operation of this spark ignited is introduced while describing the engine actuator system.

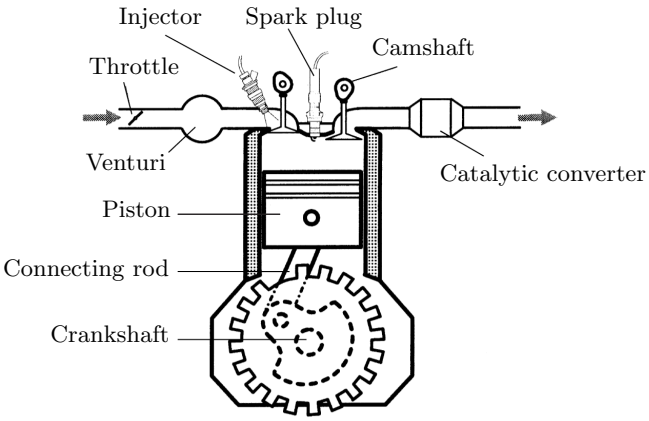


Figure 7.8: *Basic structure of a four-stroke spark-ignition engine.*

Actuator System

The operation conditions of the considered type of combustion engines can be influenced with the actuators drawn in Figure 7.8. Fresh air is brought to the cylinders through the throttle plate, the venturi and the intake manifold. At the end of the intake manifold, the fuel is injected and vaporizes into the fresh air.
The first stroke down (intake stroke) pulls this fuel-air mixture into the combustion chamber. In the second stroke up (compression stroke), all valves are closed and the mixture is compressed. The third stroke down (power stroke) comes about through the rapid burning of the compressed fuel mixture. One might think that the fuel-air mixture explodes after ignition, but further investigation showed that it rapidly burns. Finally, the fourth stroke up (exhaust stroke) expels the exhaust gases from the cylinder. It entire process is also called the "Otto cycle".
In Figure 7.9, the working cycle of a four-stroke combustion engine is depicted. After the fuel-air mixture has been pulled into the cylinder (a) and the valves have been closed, the mixture is compressed. Shortly before the top dead center position (TDC) of the piston (b), the mixture is ignited. The time of ignition is defined as the relative angle of the crankshaft before

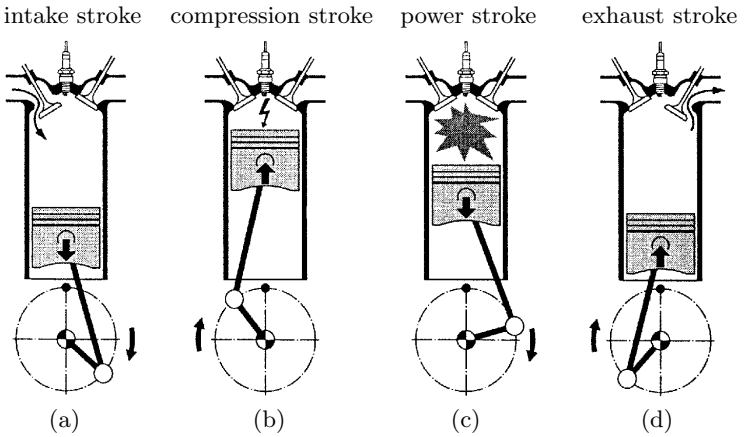


Figure 7.9: *Working cycle of a four-stroke combustion engine.*

the TDC position of the piston. From the volume V_h, when the piston resides in the bottom dead center (BDC), and the volume V_c, when it resides in the TDC, the compression ratio $\varepsilon = (V_h + V_c)/V_c$ can be determined. Depending on the construction parameters of the engine, ε adopts values between 7 and 13. When the compression ratio is increased, the thermal efficiency increases and the fuel can be used more effectively. Unfortunately the compression ratio ε cannot be infinitely increased, since a high ε also increases the probability of knocking. Knocking is a condition, accompanied by an audible noise, that occurs when the fuel-air mixture in the cylinders is ignited too early, e.g. due to high pressure or temperature. Long time knocking damages the cylinder and can destroy the entire engine.
After the compressed mixture has been ignited (c), the temperature increases due to ongoing combustion. The increasing in-cylinder pressure forces the piston downwards. The kinetic energy of the piston is transferred to the crankshaft the via the connecting rod which is then available to the transmission at the power take off. When the piston moves up again (d), the exhaust gas is expelled from the combustion chamber.

In order to exploit the occurring oscillations for faster gas exchange, the intake is starting to open while the exhaust is not yet closed. This is called valve overlap.

The throttle which is mounted on the venturi regulates the mass of air supplied to the cylinders. In modern motor management systems, the throttle plate is operated electrically which allows the engine control unit (ECU) to control the flow of air into the cylinder directly. For this the position of the driver's gas pedal is measured and the proper throttle plate angle α_{th} is computed according to the current mode of operation. From the actual air mass entering the cylinder, the ECU determines the amount of fuel to be injected into the intake manifold.

Fuel consumption, exhaust emissions and smooth operation of the engine are directly correlated to the air-fuel ratio (A/F ratio). The A/F ratio is defined as the mass of air supplied to the engine divided by the mass of fuel supplied in the same period of time. The chemically ideal, or stoichiometric, air-fuel ratio defines the exact mass of air which is necessary to burn all the carbon and hydrogen in the fuel to carbon dioxide and water with no oxygen remaining. For spark-ignition engines, an ideal combustion is achieved with an A/F ratio of $14.7 : 1$.
A measure for the deviation of the actual from the theoretical A/F ratio is the so called λ-value. This variable is defined as the quotient of the actual mass of air and the theoretically required mass of air for ideal combustion. Hence, for $\lambda = 1$ the actual mass of air is equivalent to the theoretically required mass. For $\lambda < 1$ less oxygen than required is present. An increased power output is obtained for $\lambda = 0.85$ to $\lambda = 0.96$. For $\lambda > 1$ more oxygen than required is present. A decreased fuel consumption can be observed for $\lambda = 1.05$ to $\lambda = 1.3$. Finally, for $\lambda > 1.3$ the fuel-air mixture is hard flammable and failure or misfiring can be observed.

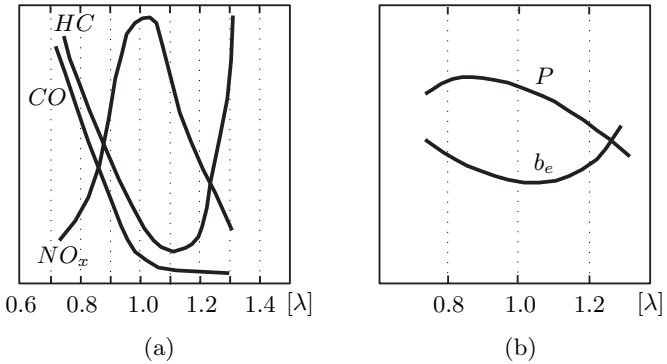


Figure 7.10: *(a) Qualitative influence of λ on the exhaust emissions. (b) Qualitative influence of λ on the power output P and the effective fuel consumption b_e of the engine.*

In Figure 7.10(a), the relative influence of λ on the power output P and the effective fuel consumption b_e of the engine is depicted. Figure 7.10(b) displays the relative influence of λ on the composition of the exhaust emissions. A modern spark-ignition engine reaches its maximum power output for $\lambda \approx 0.90$, while the minimum fuel consumption requires $\lambda \approx 1.15$ [Bos94].
The influence of λ on the composition of the exhaust emissions has to be considered separately for each type of the emitted noxious gases. As depicted in Figure 7.10(b), the concentration of carbon monoxide (CO) and the unburned hydrocarbons (HC) decreases with growing values of λ. The concentration of HC rapidly increases again for $\lambda > 1.2$. Unfortunately, the concentrations of NO_x behave totally different. The oxides of nitrogen adopt a global maximum for $\lambda \approx 1.0$ while the concentration of all other exhaust gases is relatively low and vice versa.

The exhaust emissions of a combustion engine can be manipulated in three ways. The first possibility is the preparation of the fuel-air mixture before entering the engine. The second possibility is the optimization of the combustion through a particular shape of the combustion chamber itself. Finally, the exhaust emissions can be manipulated by placing catalytic converters in the exhaust pipe of the engine. The catalytic converter improves

the oxidation of CO and HC to nontoxic CO_2 and H_2O. At the same time it reduces the dangerous oxides of nitrogen (NO_x) to harmless nitrogen (N_2).

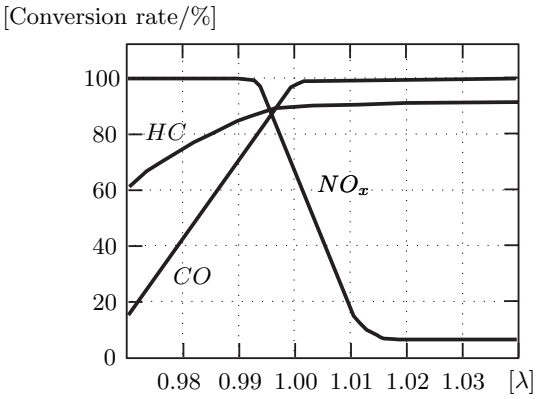


Figure 7.11: *Conversion rates of the catalytic converter.*

Figure 7.11 depicts the conversion rate of a three-way catalytic converter for each component of the exhaust gas mixture. With the aid of catalytic converters, up to 90% of the produced noxious gases can be converted into harmless substances [Bos94]. An effective catalytic conversion requires an almost optimal fuel-air mixtures with $\lambda \approx 1.0$. A deviation of 2% from $\lambda = 1.0$ yields an unacceptable conversion rate of at least one component. In order to guarantee good operation conditions of the catalytic converter, modern engines are equipped with a closed loop λ-control. The task of a closed loop λ-controller is to obtain and maintain a stoichiometric fuel-air mixture with $\lambda \approx 1.0$.

Automotive Sensor System

In order to determine the appropriate actuator settings with respect to the driver's demands and the current state of the entire system, the engine control unit is dependent on a large number of sensor signals. For the acquisition of the required data, modern combustion engines are equipped with numerous sensors. The type and localization of the sensors will be explained in this section.

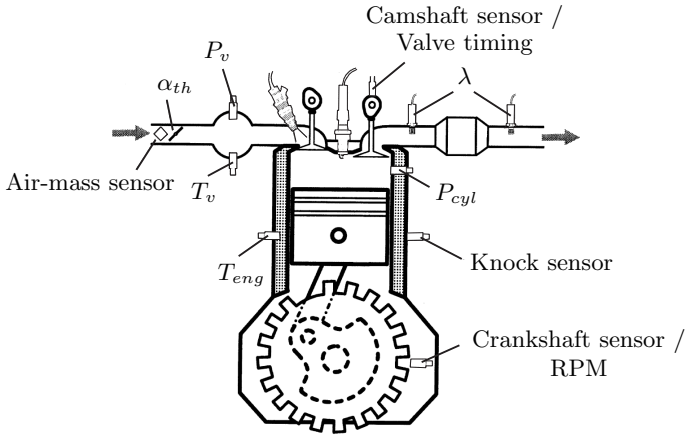


Figure 7.12: *Sensor system of a spark-ignition engine.*

Figure 7.12 depicts the basic sensor system of a spark-ignition engine together with their mount points on the engine.

- The air-mass sensor is located between the air cleaner and the throttle plate. As described in detail in [Bos94], a platinum-film resistor is brought into the air stream. An integrated circuit is keeping the sensor temperature at a specific value by controlling the electrical current through the sensor element. From the additional electrical current to maintain the sensor temperature the passing air-mass can be obtained as a measure for the actual engine load in units of kg/h. In combination with a second sensor element, also the direction of flow can be determined. This is important to identify oscillations in the venturi. In most operation conditions, the delivered signal is very

accurate. However, due to strong oscillations in some conditions the air-mass cannot be determined with this sensor signal.

- After the air-mass sensor the fresh air passes the throttle. As mentioned before, the engine control unit translated the driver's demands into the throttle angle α_{th}. The throttle regulates the flow of fresh air into the cylinders. The throttle angle is measured with a potentiometer which is mounted on the throttle shaft. The resulting signal has very high dynamics, but tents to be rather inaccurate in static situations. In modern motormanagement systems, this sensor signal is only used in dynamic situations or failure conditions.

- The third sensor to determine the air-mass is the absolute pressure sensor which is mounted in the venturi before the intake manifold. The pressure P_v in the venturi is measured very accurately with a piezoelectric resistor. This sensor shows a very good static and dynamic characteristic. In order to determine the exact air-mass flow into the intake manifold, the ambient pressure is required. Unfortunately, due to cost efficiency, an ambient pressure sensor is not present in current sensor systems. Since the air-mass flow is very hard to determine solely from P_v, dynamic system models are employed to estimate the air-mass from known sensor signals.

- In order to correct the determined air-mass with respect to its density, the temperature T_v in the venturi has to be known. The sensor element is a temperature sensitive resistor. In this case, a negative temperature coefficient resistor (NTC) is employed.

- The temperature T_{eng} of the cooling system determines the current operation condition of the engine. This sensor signal allows the ECU to distinguish whether a cold start has been performed or the engine is in the warm-up.

- The determination of the exact crank angle position of the piston is elementary for an optimal inflammation of the fuel-air mixture in the cylinder. As depicted in Figure 7.12, a toothed wheel made of ferrous metal is mounted on the crankshaft. When the crankshaft is revolving, the teeth are passing nearby a field coil wrapped around an permanent magnet. When a tooth is passing the sensor, it causes a variation of the magnetic flux and thus a measurable alternating electrical current. The start of the measurement is marked by two missing teeth, as shown in Figure 7.12. Since the frequency of the generated alternating current depends on the time gap between the

teeth, the revolution speed of the crankshaft can also be determined from this signal.

- Since two revolutions or 720° are required for a complete Otto-cycle, the true position of the crankshaft and thus of the piston cannot be determined solely from the signal of the crankshaft sensor. This is the reason for the existence of another sensor measuring the rotation of the camshaft which turns at half the speed of the crankshaft. For this, a permanent magnet is mounted on the camshaft which induces a signal in the Hall-sensor every time the camshaft completes a full revolution. Hence, the unambiguous position of the piston can only be determined from the crankshaft- in combination with the camshaft signal.
  This sensor signal is also used for the determination of the camshaft position. This is important when adjusting the position of the camshaft relative to the crankshaft so that the valves will open and close at the proper time.

- It has been already mentioned in Section 7.2.1.1, that the current λ-value has a major influence on the exhaust emissions. Since the λ value can only be determined very inaccurately from the measured air-mass and the amount of injected fuel, an oxygen sensor is placed in the exhaust pipe. The ECU determines the quality of the fuel-air mixture from the concentration of oxygen in the exhaust gas. Modern three way catalytic converters can only achieve their maximum conversion rates with this sensor. The second oxygen sensor is mandatory for the diagnosis of the catalytic converter system. Note that a minimum temperature of $200°C - 350°C$ is required for the proper operation of this sensor.

- In order to prevent the engine from mechanical defects a sensor is mounted on the cylinder block to detect the high-frequency vibrations caused by detonation or advanced ignition. Since an engine gives the best power and efficiency as it approaches detonation, the knock sensor can relay this information to a computer which then controls the factors leading to detonation. The ignition timing is the major factor leading to detonation or uncontrolled combustion. If detonation is detected, the ignition timing is immediately retarded.

- After this short presentation of the basic sensor system an additional sensor has to be mentioned which is not yet employed in series production. A piezo-resistive pressure sensor can be brought directly into

the combustion chamber for measuring the in-cylinder pressure during combustion. This pressure data will be used later in Section 7.2.4 to determine a central parameter of combustion, the so called 50% energy conversion point. A more detailed introduction to this type of sensor and the corresponding signal preprocessing can be found in [Kur94] and [Her94].

7.2.1.2 Combustion Process

In this section, the basic chemical reactions of the internal combustion process are presented. Furthermore, the relevance of the in-cylinder pressure for the analysis of the combustion process is motivated. Finally, the influence of ignition timing on the efficiency of combustion is presented.

Chemical Reactions

The fuel of a combustion engine mainly contains chemical compounds called hydrocarbons (C_xH_y). They are made of the chemical elements hydrogen (H_2) and carbon (C). Those hydrocarbons have the ability to perform a chemical reaction with the oxygen of the air. During any thermal combustion, the following principle reactions take place:

$$\begin{aligned} 2\,H_2 + O_2 &\rightarrow 2\,H_2O \\ 2\,C + O_2 &\rightarrow 2\,CO \\ 2\,CO + O_2 &\rightarrow 2\,CO_2. \end{aligned} \tag{7.2}$$

If a mixture of small- and middle chain hydrocarbons (C_xH_y) is employed, the chemical reaction with the oxygen of the air can be formulated in the following way:

$$C_xH_y + \left(x + \frac{y}{4}\right) O_2 \rightarrow x\,CO_2 + \frac{y}{2}\,H_2O. \tag{7.3}$$

If a stoichiometric fuel-air mixture and an ideal combustion is assumed, only carbon dioxide and water vapor is produced. In reality, however, the fuel is comprised of a large variety of hydrocarbons (hexane, octane, nonane), cyclic hydrocarbons (phenol, toluene) and organic sulfur compounds. Due to the various load conditions of the engine, an ideal fuel-air mixture cannot be guaranteed for every combustion cycle. Since the air is

not comprised of pure oxygen but of a mixture of oxygen and nitrogen, also oxides of nitrogen are produced during the combustion process. Finally, the combustion itself is not ideal in reality. This leads to the fact that the exhaust emissions of a combustion engine is mainly a mixture of unburned hydrocarbons (HC), oxides of nitrogen (NO_x), carbon monoxide (CO), carbon dioxide (CO_2) and water vapor (H_2O). The interested reader is referred to [PKTS89] for a more detailed description of the chemical processes.

Combustion Pressure and Energy Conversion

Since the in-cylinder pressure in an integral variable of all effects influencing the combustion, this signal is most suitable for the analysis of the combustion process. As described above, very robust sensors exist which can be brought directly to into the combustion chamber through an additional drill hole in the cylinder head.

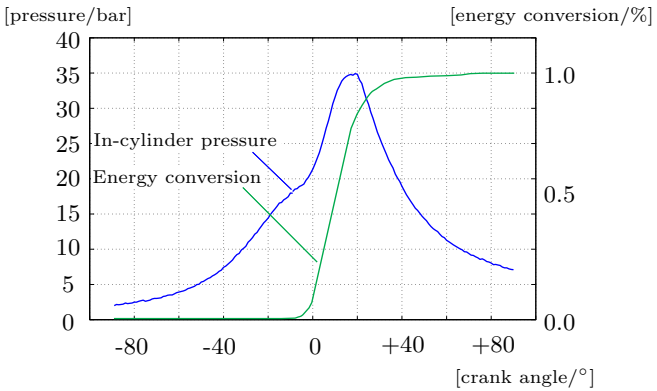


Figure 7.13: *In-cylinder pressure and energy conversion.*

Figure 7.13 depicts the in-cylinder pressure curve during combustion. The crank angle is measured relatively to the top dead center position of the piston at the end of the compression stroke. After the fuel-air mixture has been pulled into the cylinder, the mixture is compressed and ignited shortly before the piston reaches the top dead center position. After ignition the

pressure increases rapidly until it reaches its maximum value shortly after the top dead center. The resulting in-cylinder pressure accelerates the piston on its downward movement. The accelerating force is transferred to the transmission system via the connecting rod and the crankshaft.

From the mere in-cylinder pressure curve it is very difficult to determine valuable information about the quality of the performed combustion directly. In order to obtain immediate information about the combustion process, the energy conversion curve is considered. The energy conversion curve can be computed from the measured combustion pressure through a sophisticated, recursive thermodynamic calculation. In Figure 7.13, the in-cylinder pressure and the corresponding energy conversion curve is depicted.
Besides the λ-value, the ignition timing has also a significant influence on the produced exhaust emissions and the fuel consumption. It is important that the spark coming from the spark plugs ignites the air-fuel mixture when the crankshaft is in the correct angular position. If the ignition timing is advanced, the concentration of unburned hydrocarbons increases due to incomplete combustion of the fuel-air mixture. In that case, it can also be observed that the temperature in the combustion chamber increases. This results directly in a higher concentration of NO_x in the exhaust gas. Only the emission of CO is almost independent of the ignition timing.

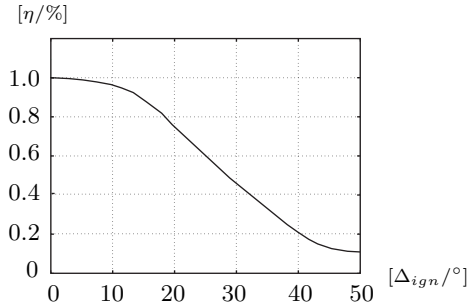


Figure 7.14: *Efficiency of a combustion engine versus the deviation from the optimal ignition timing.*

Figure 7.14 depicts the efficiency of a combustion engine as a function of the deviation from the optimal ignition timing. It can be observed that small deviations around 10° from the optimal ignition timing have only minor effects on the efficiency. On the other hand, a deviation above 10° results in severe degradation of the engine efficiency.
In BARGENDE [Bar95a], a new criterion for the analysis of the engine efficiency with respect to the ignition timing has been presented. It turned out that the so called 50% energy conversion point (50%-ECP) of the current combustion is of major interest. The 50% energy conversion point is defined as the crank angle position at which 50% of the fuel mass in the cylinder has chemically reacted with the oxygen of the air. BARGENDE found out that when the engine works under condition of medium load, the 50%-ECP should be at approximately +8° after the TDC of the piston in order to achieve maximum engine efficiency. Hence, the ignition timing has to be adjusted appropriately to guarantee this outcome. In further investigations, BARGENDE [Bar95b] showed clearly that the 50%-ECP is a simply but effective criterion to optimize and maintain the efficiency of a combustion engine during operation. As mentioned before, due to computational complexity it had not been possible to determine the 50%-ECP in a real-time environment. In Section 7.2.3, a method for the fast computation of the 50%-ECP with GMI-optimized neural networks will be presented.

7.2.1.3 Structures of Modern Motormanagement

Since the first use of electronic components for the management of combustion engines, a remarkable development of the employed hardware could be observed. This section intents to present a brief insight into the complex soft- and hardware structures of motor management.

Engine Control Unit

Compared to the first rather simple electronic circuits, modern engine control units are powerful computers with dedicated real-time operating systems. The electronic control unit is the central processing unit of the motormanagement system.
Figure 7.15 depicts the functional structure of a modern engine control unit. Switched input signals, e.g. the selected gear, are directly fed into the CPU after passing an appropriate signal conditioning circuit. Analog input sources, e.g. the signal from the knock detection sensor, are converted into digital values before entering the CPU. On the basis of the collected

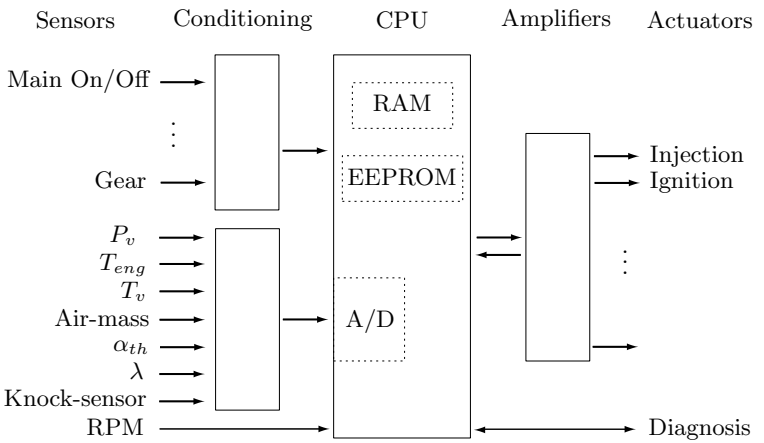


Figure 7.15: *Functional Structure of an Engine Control Unit.*

input signals from the sensor system, the CPU calculates the appropriate actuators settings.

To perform all tasks of signal processing, the CPU requires an executable program code. This code is stored together with relevant variables and characteristic maps in the EEPROM of the engine control unit. The determined output signals are then applied directly to the connected actuators via power circuitry. If any error occurs during operation, it is stored into the EEPROM and can be read out through the serial diagnosis interface at the next service interval.
During development, almost every function can be adjusted to the requirements of a particular engine type with a large number of parameters and characteristic maps. Due to the immense complexity of a modern ECU, the determination of this characteristic maps requires several years of man power. Note that the parametrization of the implemented functions with respect to only one particular engine consumes a very large amount of the entire development budget.

In order to integrate other electronic control devices, e.g. the anti-lock brake system (ABS), anti-spin regulation (ASR) or the board computer in-

side the passenger compartment, an efficient communication among those devices is necessary. Since the wiring harness in a modern passenger car has grown extremely complex, conventional point-to-point connections are not efficient enough to provide the required bandwidth for the communication of all essential signals. This problem has been solved with the introduction of the controller area network (CAN). The CAN is a digital bus system with collision avoidance mechanism. For a detailed description of the CAN bus system the interested read is referred to [Law97] or [Bos94].

Conventional Ignition Timing

As already mentioned above, the ignition timing of a combustion engine depends on the engine load and the engine speed. This variables are measured with the corresponding sensor elements and transferred to the ECU. Since the environmental conditions and the operating modes of the engine are rapidly changing, the ignition timing has to be constantly corrected to meet the requirements of new circumstances.

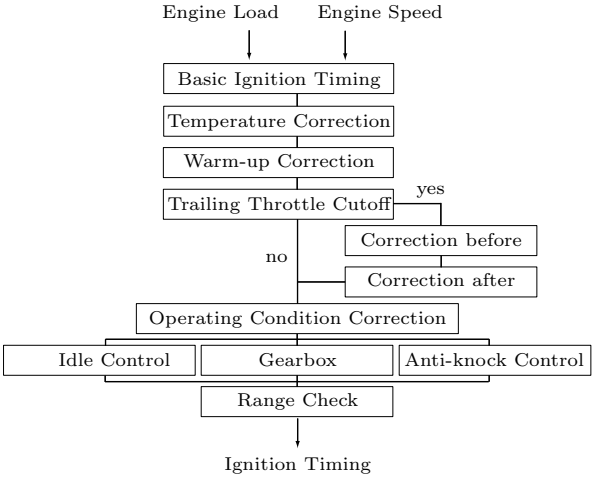


Figure 7.16: *Algorithm in the engine control unit which determines the proper ignition timing.*

Figure 7.16 depicts, according to [Bos94], the conventional algorithm to compute the proper ignition timing. After the basic ignition timing has been determined from a static map inside the ECU, this value has to be corrected to guarantee proper ignition for a large variety of operating conditions. An example for a static map containing the basic ignition timing is depicted in Figure 7.17. In the start-up phase and depending on the ambient temperature, corrections are applied to the basic ignition timing. In special operating conditions, e.g. when starting or leaving the trailing throttle fuel cut-off, the ignition timing has to be adjusted also to the requirements of other vehicle components. Finally, if the engine is detected to run in idle-state or if knocking combustion is detected, a closed loop controller is activated which takes over the adjustment of the ignition timing.

In this section, a brief overview of the complex soft- and hardware structures in modern motormanagement systems has been provided. The next section intents to introduce the basic principles of neural combustion control.

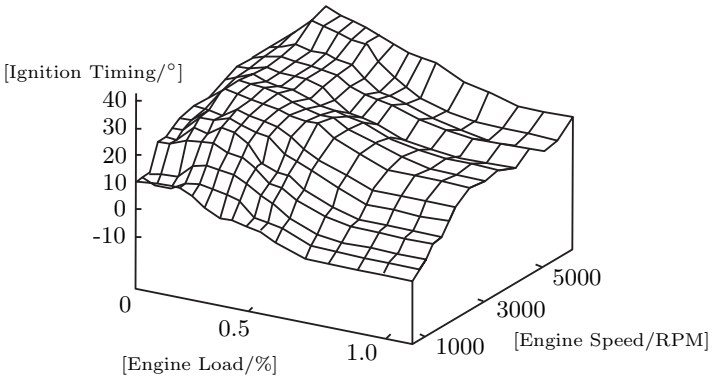


Figure 7.17: *Characteristic map of the basic ignition timing.*

7.2.2 Neural Combustion Control

Modern motor management systems rely on numerous sensor signals, in order to be able to accomplish their tasks. From the viewpoint of combustion control, the 50% energy conversion point (50%-ECP) is of particular interest. Since this variable cannot be measured directly, it has to be determined from a measurable quantity which characterizes the quality of the observed combustion. Since the in-cylinder pressure is the central quantity which describes the effects influencing the entire combustion, this measurable quantity is used to determine the desired 50%-ECP.

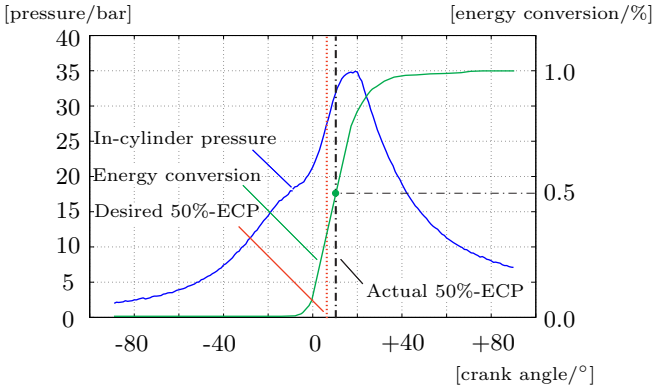


Figure 7.18: *In-cylinder pressure, the 50%-ECP and the course of the energy conversion during one combustion cycle*

Figure 7.18 displays the in-cylinder pressure curve and the according energy conversion as a function of the crank angle position during one combustion cycle. The combustion pressure is the central variable with respect to all effects influencing the course of combustion. It is used to determine the 50%-ECP from which the efficiency of the observed combustion can be estimated.
The 50% energy-conversion point is defined as the crank angle position at which 50% of the fuel mass in the cylinder has chemically reacted with the oxygen during the course of combustion. In case of an optimal combustion with maximum efficiency, the position of the 50%-ECP has to be at a crank angle position of approximately $+8^\circ$ after the top-dead-center position (TDC) of the piston.

It has been shown earlier by BARGENDE [Bar95b] that the efficiency of a combustion engine is maximized with the 50%-ECP around +8° TDC (±2°). Compared to other energy conversion points, the 50%-ECP stays relatively stable throughout all operation conditions of the engine. In order to keep the 50%-ECP at the desired crank angle position which depends on the current load level of the engine, a closed loop PI controller could be implemented.

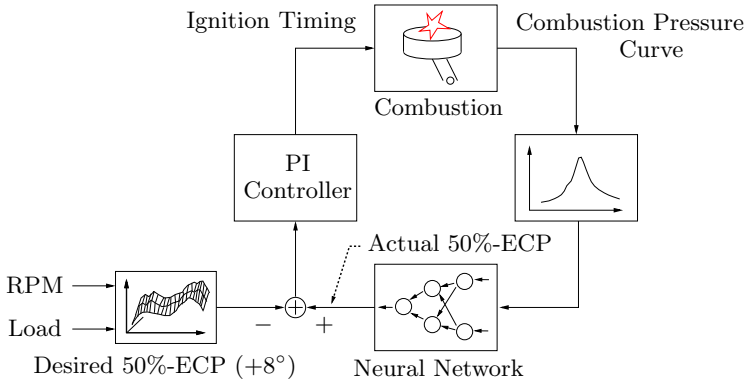


Figure 7.19: *Basic structure of a closed loop neural combustion control.*

Figure 7.19 depicts the basic structure of a closed loop combustion controller. As mentioned before, the goal of combustion control is to keep the 50%-ECP at a desired crank angle position. This crank angle position depends on the actual operating condition of the engine. Referring to BARGENDE, it could be either set to +8° TDC or it could be determined via a static map inside the engine control unit (ECU). The in-cylinder pressure is constantly measured during the course of combustion and the crank angle position of the 50%-ECP is determined from the obtained pressure curve. The deviation of the current 50%-ECP from the desired value is fed into a PI controller, which then adjusts the ignition timing for the next combustion cycle.
The 50%-ECP might be obtained with a high degree of accuracy from the measured pressure curve through a sophisticated thermodynamic computation. Since this calculation is far too complex to be accomplished in real time, feedforward neural networks have been proposed as an alternative

approach. In this attempt, the target output of the neural structure is obtained through the mentioned thermodynamic computation before the actual training is performed. This approach takes advantage of the capability of neural networks to learn and generalize complex nonlinear functions. Once training has been successfully completed, the neural network can be regarded as a virtual sensor for the 50%-ECP. Finally, it will be integrated into the control structure shown in Figure 7.19.
At the end of this chapter, it will also be shown that this method is suitable for real time applications and can thus be implemented in automotive environments.

In order to determine the actual 50%-ECP with a neural network, the in-cylinder pressure is equidistantly sampled and fed into the neural structure after training. The basic requirement with respect to the maximum deviation of the derived virtual combustion sensor, from the effective 50%-ECP, was specified to be $\pm 2^\circ$.

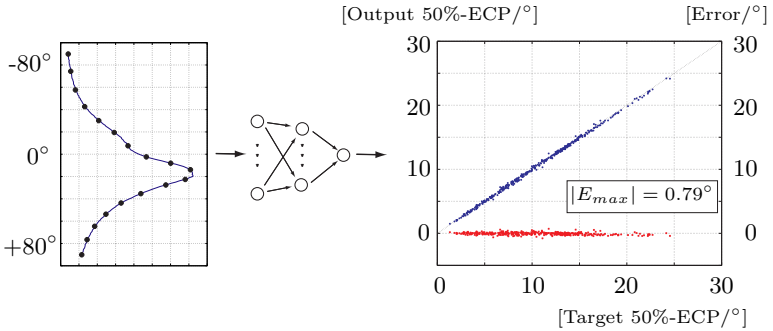


Figure 7.20: *Equidistant down-sampling of the in-cylinder pressure and presentation to the neural network.*

Figure 7.20 shows the presentation of a 18 dimensional input vector, which is comprised of the equally sampled combustion pressure curve. The original in-cylinder pressure curve has been recorded with a resolution of 1° and contains 180 points. In this example, a 18-10-5-1 feedforward neural network has been trained with the Extended Kalman Filter for 100 epochs to map the vector of pressure values to its according 50%-ECP. The test data set contains 400 combustion cycles and the absolute value of the max-

imum error is 0.79°. However, the choice of equidistant sample-points has been an empirical decision. In spite of this remarkable result, we have to raise the following fundamental questions concerning the construction of the neural network training data.

1. Are the chosen sample points reasonable from the viewpoint of information theory?

2. Can the dimension of the input pattern to the neural network be reduced, in order to optimize the computational expenses?

3. Finally, what are the sample points with the highest information content with respect to the given problem and how can they be systematically identified?

The answers to this interesting questions will be provided in the context of the next section. Beyond this, the applicability of the optimized virtual combustion sensor in a real time environment will be proved.

7.2.3 Selection of the relevant Input Variables employing Forward Selection and the GMI I_2

In Chapter 5, the general mutual information has been introduced as a method for measuring the linear- and nonlinear dependencies among multidimensional time series. Due to the existence of an efficient estimation algorithm, the presented concept is based upon the second order Rényi-Entropy I_2. The practical aspects of the algorithm included the introduction of an adequate matrix calculus and the assessment of its computational complexity.

The question of interest is now whether it is possible to employ the presented method for the analysis of real time measurement data in general. In this chapter the general mutual information will be employed for the preprocessing of neural network input data in an automotive environment. It will be demonstrated how the most relevant sample points can be selected from a large set of combustion pressure curves to construct optimal neural network training data [HS00]. Besides the employment of the GMI method for spark ignition engine data, the remarkable broad applicability of this sophisticated method proved later to be also of great value for the analysis of common-rail signals in diesel engines [Frö03].

The questions at the end of Section 7.2.2 can be quantitatively answered by employing the previously introduced concept of the general mutual information I_2. In order to construct an optimized data set for neural network training, the input variables with the highest information content, with respect to a given problem, are iteratively selected from a large data set.

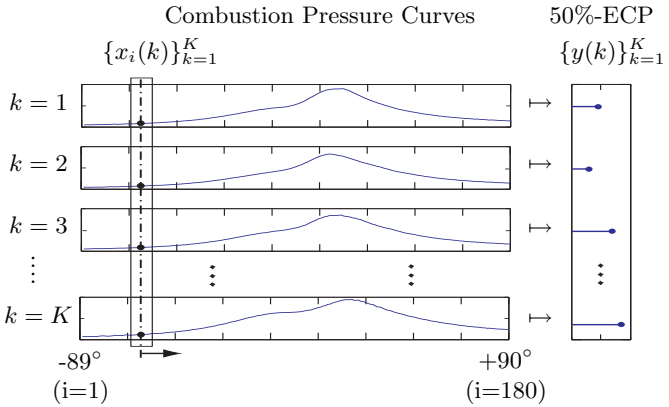


Figure 7.21: *Calculation of the GMI between i-th sample point of the combustion pressure curves* $\{x_i(k)\}_{k=1}^K$ *and the corresponding 50%-ECPs* $\{y(k)\}_{k=1}^K$.

In Figure 7.21, the sampled combustion pressure curves $\{x_i(k)\}_{k=1}^K$ and its according 50%-ECPs $\{y(k)\}_{k=1}^K$ are depicted. As mentioned above, the most relevant sample points of the combustion pressure curve are identified through an iteration process. For this purpose, the i-th pressure values $\{x_i(k)\}_{k=1}^K$ in all K pressure curves together with the 50%-ECPs $\{y(k)\}_{k=1}^K$ are considered. A quantitative measure for the relevance of this particular sample point for the determination of its corresponding 50%-ECP is obtained by computing the general mutual information I_2 between the sequences $\{x_i(k)\}_{k=1}^K$ and $\{y(k)\}_{k=1}^K$. The resulting GMI values for $i = 1, ..., 180$ are then normalized and plotted against the crank angle position.

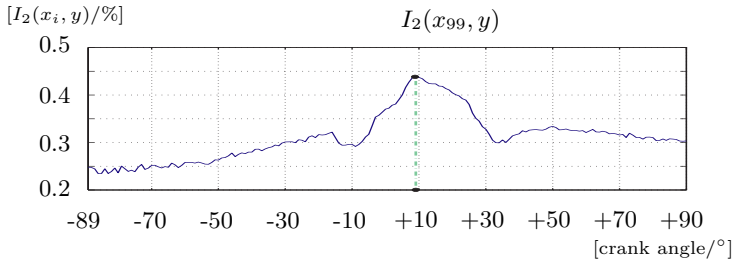


Figure 7.22: *The GMI function reaches its maximum at the crank angle position with the highest information content.*

Figure 7.22 depicts the GMI values of the first iteration which are plotted against the crank angle position of the piston. The position showing the highest GMI value is interpreted as the most relevant sample point of the pressure curve for determining the 50%-ECP. In this case, the sequence $\{x_{99}(k)\}_{k=1}^{K}$ corresponding to the crank angle position $+9°$ has been identified as the most relevant sample point of the underlying combustion pressure curves.

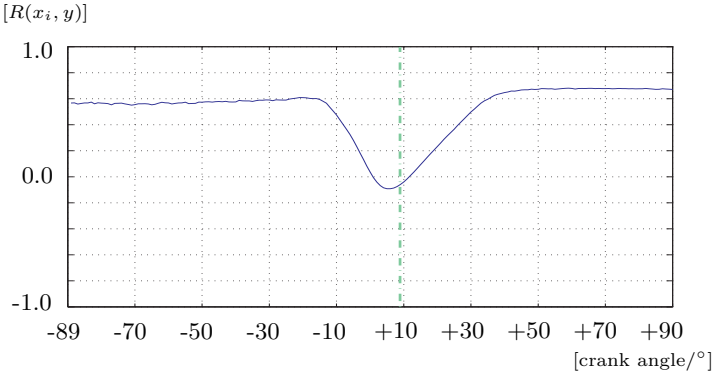


Figure 7.23: *Spearman's rank correlation coefficients adopt values around zero for crank angle position where the GMI reaches its maximum.*

For one dimensional time series, the coefficients of correlation $\rho(x_i, y)$ can also be computed and compared to Figure 7.22. The rank correlation coefficients are based on the absolute ranks of the observations. This statistic measure does not depend on a specific distribution of the underlying variables and is called a *nonparametric-* or *distribution-free statistic*.
In Figure 7.23, it can be clearly observed that the rank correlation coefficients [Spe04] adopt values around zero whereas the GMI in Figure 7.22 reaches its maximum. Hence, no variable would be chosen which corresponds to crank angle positions in the range from $-20°$ to $+40°$. Since this range depicts the so called high-pressure phase, i.e. the crank-angle positions at which the actual combustion occurs, this result does clearly not contribute to the solution of our selection problem.

The GMI function in Figure 7.22 might also remind of the coefficient of correlation. However, the general mutual information differs from the coefficient of correlation in multiple points.

- The GMI is defined also for time series of arbitrary dimensionality, while the coefficient of correlation is not.
- All linear- and nonlinear dependencies between the possible multi dimensional sequences are captured by the GMI. The conventional coefficient of correlation is only sensitive to linear dependencies.
- If the GMI value tends to zero, it can be concluded that there are no statistical dependencies between the investigated time series (Theorem 5.3). From a coefficient of correlation of zero no conclusion can be drawn at all, for arbitrarily distributed time series.

According to the forward selection strategy mentioned before, all previously identified points have to be considered. The additional sequence corresponding to the next chosen sample point, has to maximize the obtainable information of the given problem.
The second iteration of the selection process is illustrated in Figure 7.24. Similar to Figure 7.21, the GMI function is computed for each crank angle position. The GMI values are now computed between the two dimensional input sequence $\{x_i(k), (x_{99}(k))\}_{k=1}^{K}$, for $i = 1, ..., 180$, and the one dimensional output sequence $\{y(k)\}_{k=1}^{K}$ corresponding to the 50% energy conversion points.

Figure 7.25 depicts the result of the second iteration of the selection process. Compared with $I_2(\{x_i\}, y)$, the GMI function $I_2(\{x_i, (x_{i_1})\}, y)$ with

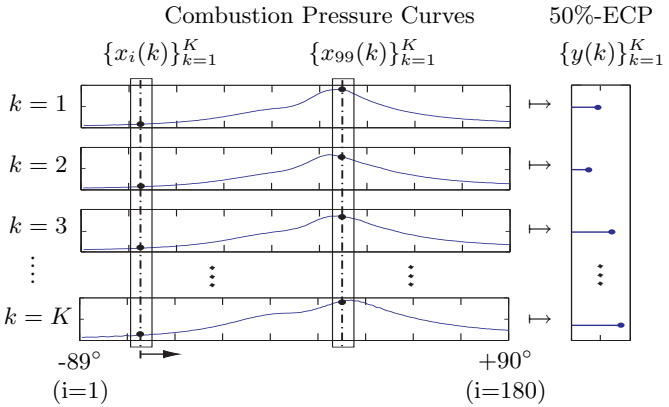


Figure 7.24: *Calculation of the GMI values between the two dimensional input sequence* $\{x_i(k),(x_{99}(k))\}_{k=1}^K$ *and the corresponding 50% energy conversion points* $\{y(k)\}_{k=1}^K$.

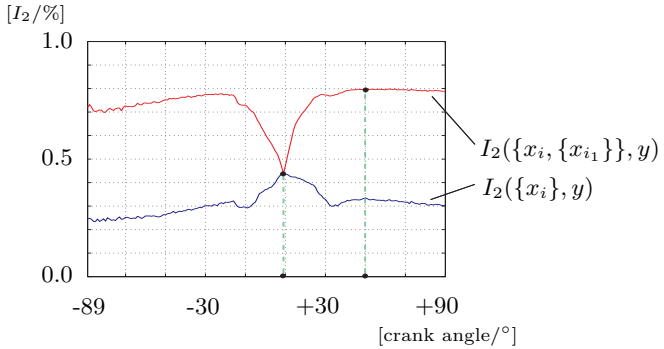


Figure 7.25: *The GMI function* $I_2(\{x_i,(x_{i_1})\},y)$ *resulting from the second iteration of the selection process.*

$i_1 = 99$ includes the information gain when increasing the dimension of the input time series by one. Regarding the GMI function $I_2(\{x_i,(x_{i_1})\},y)$, the most interesting point is when x_i reaches the formerly chosen value x_{i_1}. At this point, no additional information is present and the GMI adopts a local

minimum. The GMI value of this point is hence identical to the according point in the preceding GMI function $I_2(\{x_i\}, y)$. Continuing with this forward selection strategy, an ordered set of crank angle positions, specifying the positions with the highest information content, can be successively selected from the large set of 180 possible input variables.

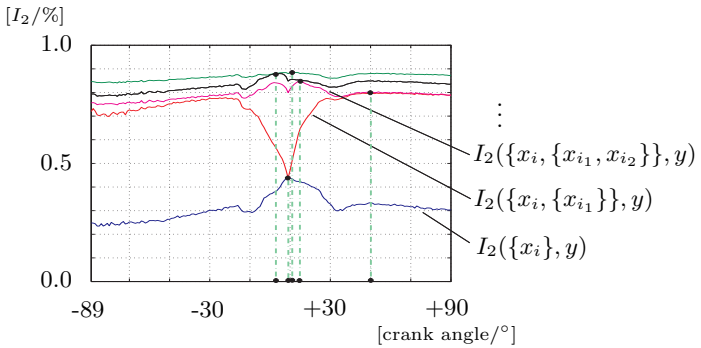


Figure 7.26: *Sequence of GMI functions for successive iteration steps of the selection process.*

Figure 7.26 illustrates the GMI functions of successive iteration steps which were used to identify the most relevant sample points of the in-cylinder pressure curve. The described procedure is carried out while the information gain is significant. When the consideration of additional input variables will not yield further information about the output variable, the iteration process is stopped. It can be observed, that the selected points are not equally distributed like previously assumed. However, from the viewpoint of information theory, the identified set of crank angle positions is indeed optimal.
Employing the concept of mutual information resulted in a significant reduction of the input dimension. The dimensionality of the input vector could be decreased from 18 down to five sample-points.
The most important result is that the previously chosen equidistant sample points are not reasonable from the viewpoint of information theory.

Finally, another interesting point is the dependence of the selected input vector components among each other. In Figure 7.27, the general mutual information $I_2(x_i, x_j)$ between pairs of input variables x_i and x_j is de-

picted. It can be observed that the resulting matrix is symmetric, since $I_2(x_i, x_j) = I_2(x_j, x_i)$ holds as mentioned in Theorem 5.3. The main diagonal consists exclusively of ones. This is also an immediate consequence of the determination property stated in Theorem 5.3. The GMI values for

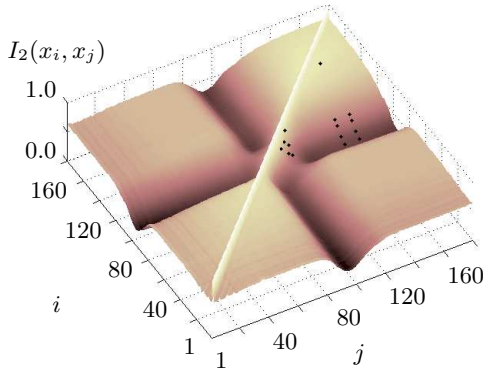


Figure 7.27: *General mutual information $I_2(x_i, x_j)$ between pairs of input variables x_i and x_j. The GMI values for the pairs of the selected input variables are outlined by dots.*

the pairs of the selected input variables are outlined by dots. It can be observed that not all selected variables are mutually independent. In the case of independent input variables, the term $I_2(x_i, x_j)$ adopts values close to zero.
If we reconsider some typical combustion pressure curves in Figure 7.21, it appears quite natural that neighboring variables have to be highly correlated due to the similarity among different in-cylinder pressure curves.

The general mutual information cannot be used to construct orthogonal input variables. Instead, it finds the most relevant input variables with respect to the nonlinear mapping between pressure values and the 50%-ECP. Hence, a further preprocessing method for neural input data such as principal component analysis (PCA) as suggested in [Bis99] could be additionally applied afterwards. This might be done to produce orthogonal input variables which might yield better convergence during neural train-

ing. However, the investigation of this fact is definitely beyond the scope of this work and might be subject to further research activities.

Compared to the conventional coefficient of correlation, the general mutual information has several major advantages:

- The mutual information is capable of capturing nonlinear dependencies between inputs and outputs.
- It can be applied to multivariate input- and output data.
- The mutual information provides a constructive method to reduce the input dimension of neural training data, on the basis of a well defined information theory.

This section was concerned with the systematic analysis of the in-cylinder pressure during combustion to determine the most relevant sample points for neural network training. The next section assesses the applicability of the obtained GMI results for a neural identification of the combustion process. Finally in Section 7.2.6, this optimized neural structure is used to implement a real-time combustion control in an experimental automotive setup.

7.2.4 Neural Identification of the Combustion Process considering the GMI Results

In Section 7.2.3, the most relevant input variables for the identification of the combustion process have already been selected. In order to prove the applicability of this result, for the concrete implementation of an optimized closed loop control, the combustion process has to be identified in advance. Since the feature selection procedure has already been completed, a neural training data set can be constructed, which comprises the realizations of the identified sample points of the in-cylinder pressure curves.

Network Topology	Input Variables	Learning Algorithm
FF-18-10-5-1	x_5, x_{10}, ..., x_{180}	RTRL, EKF
FF-5-10-5-1	x_{99}, x_{140}, x_{105}, x_{93}, x_{101}	RTRL, EKF

Table 7.3: *Neural training setups for the derivation of a virtual combustion sensor.*

Table 7.3 summarizes the neural training setups for the derivation of the virtual combustion sensor. In order to realize the impact of the GMI feature selection on the performance of the neural combustion model, two separate identification runs have been conducted. The first employs the equidistantly sampled pressure curves, while the second uses the GMI selected inputs.
Furthermore, each identification run has been accomplished with two different training methods, the Extended Kalman Filter and the Real Time Recurrent Learning algorithm. The purpose of this is the assessment a potential performance enhancement through the application of a sophisticated training algorithm like the Extended Kalman Filter. In all cases, the learning rate has been set to 0.05 and the training was conducted over 100 epoches.

The presentation of the GMI identified points of the in-cylinder pressure curve to the neural combustion model is presented in Figure 7.28. In this case, a 5-10-5-1 network has been trained with the Extended Kalman Filter for 100 epochs.

After training, the performance of the neural combustion model was assessed with a test data set containing 400 unknown patterns. The maximum

deviation of the virtual combustion sensor turned out to be 1.48°. When comparing this result with the outcome of the equidistantly sampled pressure curve in Figure 7.20, it can be observed that the maximum deviation increased from 0.79° to 1.48°. However, this deterioration of performance is vindicable, since the deviation is still below the predefined tolerance range of ±2.0°. In exchange for the drawback of a lowered accuracy, the reduction of the number of inputs from 18 to only 5 sample points, implies the extension of the real time capability of the obtained virtual combustion sensor.

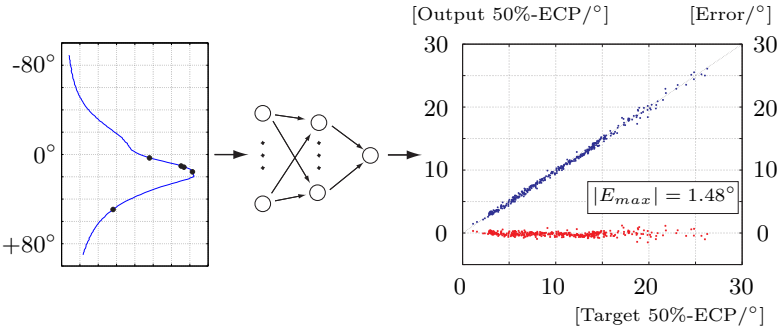


Figure 7.28: *Presentation of the GMI selected points of the in-cylinder pressure curve to a neural network after Extended Kalman Filter training.*

In Appendix E, all results of the neural training setups listed in Table 7.3 are outlined. It can be observed that the employment of the Extended Kalman Filter constantly yields to better results, compared to the RTRL algorithm. The goal was to derive a virtual combustion sensor, which is capable to determine the 50%-ECP with a maximal deviation of ±2°. At the same time, the necessary number of inputs had to be reduced to a minimum, to guarantee its computability in real time environments. It could be impressively demonstrated that both objectives can be successfully achieved by employing a prior feature selection on the basis of the GMI.

As mentioned before, the construction of an efficiently computable virtual combustion sensor is an indispensable prerequisite for its integration into a realtime capable, closed loop combustion controller. In Section 7.2.6, it

will be demonstrated how an optimized closed loop combustion controller is implemented on the basis of this virtual sensor. Finally, it will be shown how this combustion controller has been integrated into an experimental automotive setup to evaluate its performance in subsequent road tests. Prior to this, the influence of the employed feature selection method on the performance of the derived neural combustion model, is depicted and discussed in the next section.

7.2.5 Neural Identification with the GMI selected features in contrast to PCA- and ICA results

To give a detailed overview of the common state-of-the-art techniques in feature extraction, the principal component analysis and the independent component analysis have been already described in Chapter 3. In this context, a matter of particular interest is the performance of the proposed GMI method in comparison to these well established feature extraction methods. For this purpose, the mentioned methods are employed for feature extraction prior to the network training of the 50%-ECP. After successful training, the performance of the neural combustion models are compared with each other.

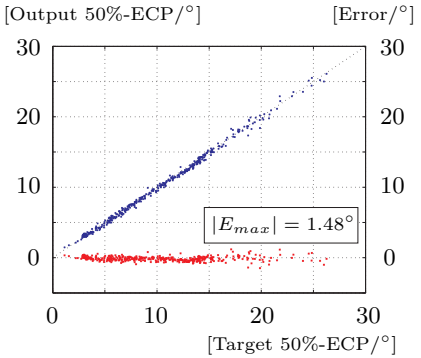


Figure 7.29: *Response of the neural combustion model after Extended Kalman Filter training of the GMI selected features of the combustion pressure curve.*

Figure 7.29 depicts the response of the virtual neural combustion sensor after learning the GMI selected features of the combustion curve. In this and in all other cases of this section, a 5-10-5-1 network has been trained for 100 epochs with the Extended Kalman Filter mentioned earlier in Section 4.3.4. It can be observed that the value of the maximum error of this constellation is 1.48° and hence meets the requirement of a maximum deviation of ±2° from the desired 50%-ECP. This outcome, which is also the basis for the implementation of the closed loop neural combustion controller in Section 7.2.6, serves as a baseline for the further comparison of the GMI- with the PCA- and the ICA feature extraction method.

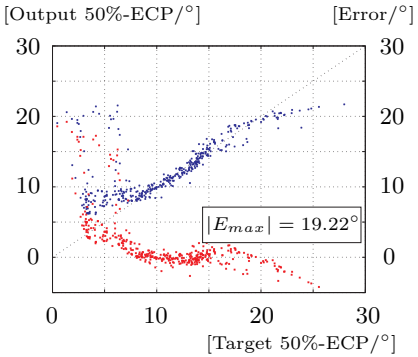


Figure 7.30: *Output of the neural combustion model after Extended Kalman Filter training of the PCA determined features.*

The response of the neural combustion model, which has been trained with the PCA extracted features, is depicted in Figure 7.30. The absolute value of the maximum error of this constellation can be observed to be 19.22°. This is by far beyond the previously mentioned acceptability threshold of ±2° with respect to the absolute deviation from the desired 50%-ECP.
The preceding principal component analysis was conducted to retain only those components which contribute more than two percent to the variance of the entire data set. Even with this realistic assumption, the PCA was only able to identify two principal features for the construction of the neural training set. This means vice versa that, according to the result of the conducted PCA, at least 98% of the data set variance are explained by these two features. Anyway, experiments showed that the number of obtained features stayed the same, even if the variance threshold was set to values below one percent. It is quite obvious that the quality requirements are not met and that the PCA is not a feasible method in this case.

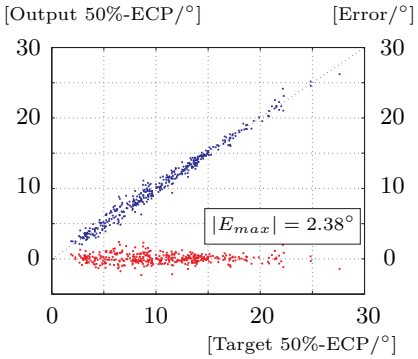


Figure 7.31: *The neural combustion model after Extended Kalman Filter training of the ICA determined features.*

Finally, an independent component analysis has been conducted on the data set and a virtual neural combustion sensor has been derived from the obtained features. Figure 7.31 depicts the output of the neural combustion model after training with the Extended Kalman Filter for 100 epochs. In order to be able to draw a comparison between the performance of the GMI selected features, the ICA was parameterized to identify also five independent components. Hence, the neural training set is comprised of five input variables, as in the case of the GMI selected features.
It can be observed that the maximum absolute deviation of this neural model is now 2.38°. This is slightly above but very close to the specified requirement of ±2° with respect to the 50%-ECP of the combustion cycle. In contrast to the performance of the model, derived from the GMI selected features, its error is of the same magnitude. Experiments showed that, in this case, the neural models with GMI selected features consistently performed slightly better than its ICA pendents. The reason for this might be found in the fact, that the GMI takes into account the input- and the output variables, whereas ICA only considers the inputs. In any case, an adequate answer to the question which of the two methods performs better will involve statistical significance tests. Since a detailed statistical evaluation is beyond the scope of this work, this might be an excellent subject for further research activities.

However, the essential outcome is that the performance of the GMI based feature selection appeared to be of the same magnitude as the performance

of the well established ICA. Apparently, it turned out that the proposed feature selection method on the basis of the GMI is a serious alternative to well established methods. While comparing the GMI based method with the principal component analysis and the independent component analysis, it became clear that the GMI is a feasible and versatile method, which can be flexibly employed in various constellations for the problem of obtaining features from considerable large data sets.
In the next section, the previously described work culminates in the final implementation of the optimized virtual neural combustion sensor in an experimental automotive setup. It will be shown, how this virtual sensor is integrated in a closed loop combustion controller and how it performed in a hard real time environment.

7.2.6 Real Time Neural Combustion Control in an Experimental Automotive Setup

This section is dedicated to the final implementation details of an optimized neural combustion control in an experimental setup on the basis of the previously presented results. The basic structure of combustion control in Figure 7.19 is used as a starting point for the implementation in a prototype car. According to BARGENDE [Bar95b], the ignition timing is set to an optimal value if the 50%-ECP of the following combustion cycle is determined to be at $+8°$. This statement is valid for all spark-ignition engines regardless of their actual operating point. Hence, the static map in Figure 7.19 can be substituted with the desired crank angle position of $+8°$. Furthermore, the existence of a linear function between ignition timing and the resulting 50%-ECP has been proved by CSALLER [Csa81]. It implies the choice of a linear controller for the adjustment of the ignition timing. This must not be confused with the dependency between the combustion pressure curve and the 50%-ECP, which is definitely nonlinear.

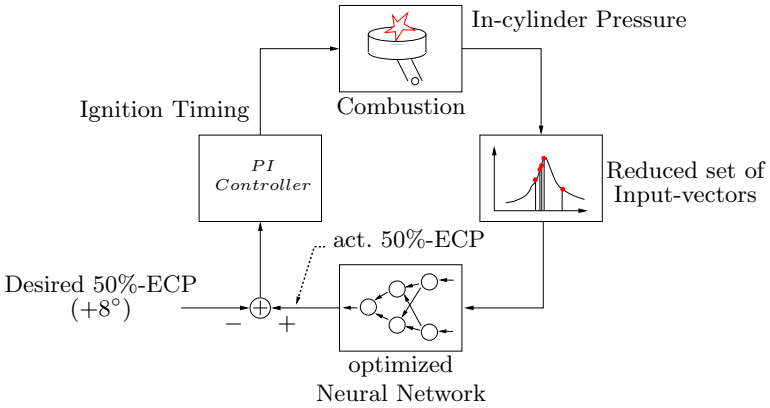


Figure 7.32: *Control-structure for the verification of the GMI analysis results. Only points with the highest information content are taken from the in-cylinder pressure curve.*

Figure 7.32 depicts the control structure which is used for the verification of the GMI analysis results obtained in Section 7.2.3. This structure has

been implemented on a prototype ECU which is comprised of two digital signal processors (DSP) mounted on a standard PC. The experimental setup was an E-Class Mercedes equipped with the 2.3 liter spark ignition engine M111E23.

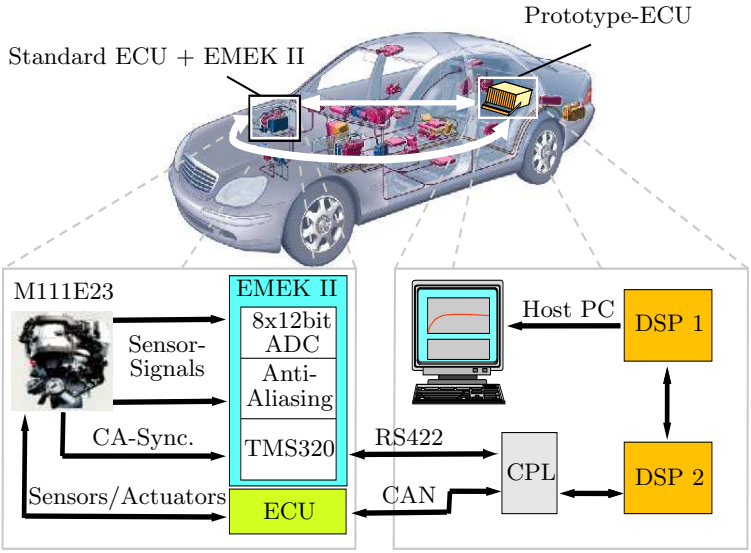


Figure 7.33: *Hardware-Structure of the measurement setup. Communication between the series- and the prototype ECU is realized with a Controller Area Network (CAN). The real time measurement system EMEK II and the prototype ECU are communicating via RS422 link.*

In Figure 7.33, the hardware structure of the employed measurement setup in the car is presented. The experimental setup is comprised of a modified series ECU, the real time measurement system EMEK II and a customized prototype ECU. The real time measurement system EMEK II is synchronized with the camshaft- and crankshaft trigger signals from the engine when recording the sensor signals. In each segment, i.e. after each combustion cycle, the data is transferred via RS422 link to DSP 2 over a Comport-

Link-Converter (CPL). The PI controller and the neural network, which is responsible for the evaluation of the 50%-ECP, are implemented on DSP 2. Finally, for the realization of a feedback control loop the new ignition timing is transmitted over the Controller Area Network (CAN) link to the modified series ECU which applies it to the next combustion cycle.
Parallel to this combustion control loop, the measurements and results are transferred from DSP 2 over the DSP communication port to DSP 1 which has direct memory access (DMA) to the host PC. The host PC is responsible for initializing the measurement system and the configuration of the DSP cluster. During operation, the in-cylinder pressure, 50%-ECP and RPM can be directly accessed through the host PC.

As mentioned before, the linearity between ignition timing and the resulting 50%-ECP has been proved by CSALLER [Csa81]. Hence, a linear controller can be used for the task of adjusting the ignition timing according to a desired value.

The employed PI controller has the following generic structure:

$$t_{ign}(k) = t_{ign}(k-1) + p\,(ECP_{desired}(k-1) - ECP_{act}(k-1)), \quad (7.4)$$

where p is the proportional element which could be specifically adjusted for each cylinder. Experiments showed that with $p = 0.5$ no unstable behavior was observable and the closed control loop showed an acceptable dynamic response. However, in certain operating conditions of the engine, the desired 50%-ECP in Equation 7.4 has to be set to values different from 8°:

- **Idle**: In idle operation, the running smoothness is negatively affected by the spark advance required to achieve a 50%-ECP of 8°. In spite of a lower engine efficiency the desired 50%-ECP is set to 15°.
- **Full load**: In full-load operation, an optimal ignition timing would result in knocking combustions. To prevent the engine from serious damage, the desired 50%-ECP is set to 15° and an additional anti-knock control is activated.
- **Throttle cutoff**: Since no fuel is injected when the engine is above idle speed and the throttle is closed, the term 50%-ECP is not defined. Under this condition the combustion control temporarily suspended and the ignition timing set according to an default ignition map.

In the course of this work, all experiments were carried out for regular operating conditions of the engine. No special conditions, e.g. cold start with catalyst heating have been taken into account.

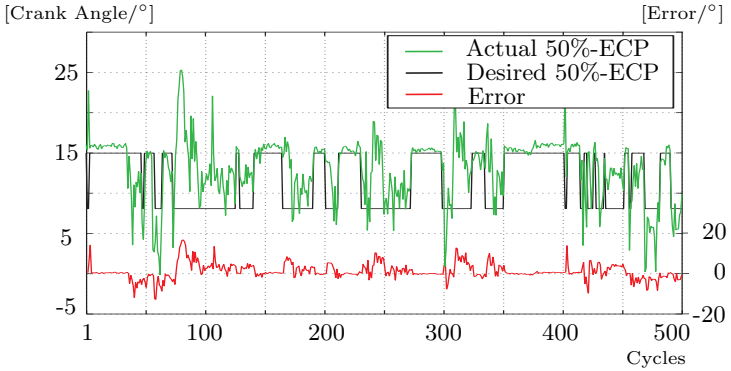


Figure 7.34: *The 50%-ECPs of cylinder 3 over 500 cycles with conventional open-loop combustion control.*

In Figure 7.34, the actual- and the desired 50%-ECP are plotted over 500 combustion cycles. In this case, the ignition angle is determined from a conventional ignition map inside the ECU as a function of engine speed and load. The deviation of the actual and the desired value of the 50%-ECP varies between $-11°$ and $+18°$.
In contrast to this, Figure 7.35 depicts the actual- and the desired 50%-ECPs of 500 cycles while combustion control is active. Compared to Figure 7.34, the actual 50%-ECP follows the desired value quite accurately with a significantly smaller deviation between $-3°$ and $+8°$.

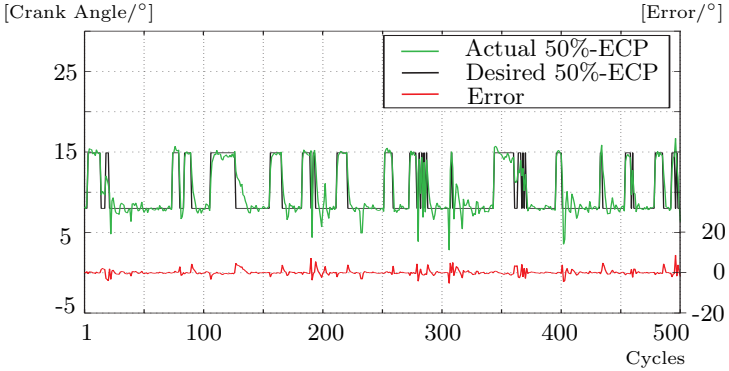


Figure 7.35: *The 50%-ECPs of cylinder 3 over 500 cycles with optimized neural combustion control.*

In this chapter, it could be demonstrated that a closed loop combustion control can be successfully implemented, based on the results of an optimized neural analysis of the combustion pressure curves. In Section 7.2.3, the mutual information has been employed for the construction of optimal neural training data with respect to the determination of the 50%-ECP. This results have further been transferred to the problem of combustion control. It could be successfully proved that an optimized neural evaluation of the combustion pressure signal is also applicable to the implementation of a combustion controller in real time automotive environments.

8 Conclusion

Since almost every interaction between entities in the real world is nonlinear in the last consequence, the employed analysis method has to account for this fact as well.
The main objective of this work was the development of a general framework for nonlinear feature selection of high dimensional data, based on the general mutual information. During the course of this work, the capability of this reputable concept of information theory has been investigated in various aspects of data analysis.

It turned out that data analysis, as far as necessary for the selection of relevant features, can be generally accomplished on the level of its implicit information content. In this context, it could be shown that the general mutual information can be successfully employed to measure the information content and hence render the selection of the most relevant features for arbitrary, high dimensional data sets possible.
Due to the fact that the general mutual information is of high specificity and sensitivity with respect to nonlinear, dynamic dependencies, it proved to be also applicable to reveal the structural- and temporal dependencies in nonlinear dynamic processes.
Furthermore it could be successfully demonstrated, that the general mutual information can be applied in various instances of feature selection strategies to identify the variables from which the actual training sets are derived. Since neural networks have been chosen as the modelling approach in this work, neural training sets are constructed as a result of the prior data analysis. Among the investigated strategies, forward selection could be identified to be most suitable for the construction of valuable training sets. Since the feature selection process is solely driven by the mutual information, the obtained data sets are definitely optimal from the viewpoint of information theory.

To demonstrate the striking interdisciplinary applicability of the information-based feature selection method in this work, it has been employed to solve problems from totally different technical disciplines. As a matter of fact, the prove-of-concept could be provided in terms of application case studies for tasks in industrial glass production and in automotive engineering.
In both cases, it turned out clearly that the dimensionality of the input space could be remarkably reduced by means of the forward selection strategy employing the general mutual information. Due to the flexible application of this method, the time dependencies in the glass melting process data could also be considered by the selection process.
The task in the automotive engineering application case, was to derive an optimized virtual neural combustion sensor from high dimensional input data. It could also be successfully demonstrated how the original 180-dimensional input space could be impressively reduced by means of this information based feature selection method [HS00] [HF01]. In a second step, this virtual sensor has been used for the construction of a closed loop combustion controller with real time capability. Finally, this hybrid controller has been implemented in a real-time automotive environment and perfectly proved to meet the required degree of accuracy.

In conclusion it can be stated that the general mutual information turned out to be a powerful and sophisticated method to objectively analyze the information content of arbitrary data sets. In combination with the forward selection strategy, it also turned out that the GMI is a credible measure for the feature selection and dimension reduction of high dimensional data. The flexible application of this sophisticated approach is also confirmed by its further employment for the different classes of problems [Frö03]. However, its range of applicability goes definitely far beyond automotive engineering or industrial glass production.
The only mentionable restriction that emerged so far when employing the general mutual information, is the mandatory availability of a sufficient amount of representative quantifiable data. In this context, the issue of missing values plays an important role. It turned out that the computation of the GMI is fairly robust for data containing less than approximately 10% missing values. In cases with more missing values, the prior application of missing-data procedures is strongly indicated.

Finally, the versatility of the general mutual information allows it to be employed in various constellations for further perspectives. When considering the performance of supervised neural training, the general mutual

information might also be employed for the online adaption of the learning rate. The modification of the learning rate during training could then be realized as a function of the mutual information between the input- and the target values of the validation data set and the according network estimates. In this constellation, the abstract flow of information through the neural structure is continuously monitored to measure the network performance during training. This, in turn, allows for the formulation of adaptive learning algorithms which incorporate the implicit properties of the training data sets.

Another interesting field for further research might be the application of the general mutual information to biometric data. This method could be of particular interest for the analysis of so called microarray data sets. Besides the standard methods like clustering and principal component analysis, the mutual information might also prove here to be a valuable method for analyzing the already high dimensional gene expression profiles.

Appendix

A Properties of $I_2(\xi, \eta)$

Proof A.1 (Properties of the GMI $I_2(\xi, \eta)$)
Statement 1 of Theorem 5.3 is determined from the symmetry of the underlying entropies $H_2((\xi, \eta)) = H_2((\eta, \xi))$.

Provided that η is uniformly distributed we get

$$\begin{aligned} I_2(\xi, \eta) &= -log \sum_m p_m^2 - log N^{-1} + log \sum_{m,n} s_{m,n}^2 \\ &= log \frac{\sum_{m,n} s_{m,n}^2}{N^{-1} \sum_m p_m^2} \\ &= log \left(1 + \frac{\sum_{m,n} s_{m,n}^2 - N^{-1} \sum_m p_m^2}{N^{-1} \sum_m p_m^2} \right) \\ &= log \left(1 + \frac{\sum_{m,n} (s_{m,n} - p_m N^{-1})^2}{N^{-1} \sum_m p_m^2} \right). \end{aligned} \tag{A.1}$$

Regarding the argument in the last line of Equation A.1, we can draw the conclusion that $I_2(\xi, \eta)$ is always non-negative. This proves the left-hand side of Equation 5.7.

$I_2(\xi, \eta) = 0$ holds if and only if $s_{m,n} = p_m N^{-1}$, for all $m = 1, ..., M$ and $n = 1, ..., N$. This fact implies that ξ and η are statistical independent. Hence, Statement 3 of Theorem. 5.3 is proved.

Furthermore the fact that

$$\sum_m p_m^2 = \sum_m \left(\sum_n s_{m,n} \right)^2 \geq \sum_{m,n} s_{m,n}^2, \quad yields$$

$$H_2((\xi,\eta)) - H_2(\xi) = -log \sum_{m,n} s_{m,n}^2 + log \sum_m p_m^2 = log \left(\frac{\sum_m p_m^2}{\sum_{m,n} s_{m,n}^2} \right) \geq 0$$

$$\begin{aligned} \iff I_2(\xi,\eta) &= H_2(\xi) + H_2(\eta) - H_2((\xi,\eta)) \\ &= H_2(\eta) - (H_2((\xi,\eta)) - H_2(\xi)) \quad \leq \quad H_2(\eta). \end{aligned}$$

$I_2(\xi, \eta) = H_2(\eta)$ is achieved if and only if for every $m \in \{1, ..., M\}$ the relation $s_{m,n^*} = p_m$ holds for exactly one $n^*(m) \in \{1, ..., N\}$, otherwise $s_{m,n} = 0$. If so, η is a function of ξ.

On the other hand we have

$$\sum_n q_n^2 = \sum_n \left(\sum_m s_{m,n} \right)^2 \geq \sum_{m,n} s_{m,n}^2$$

and hence

$$H_2((\xi,\eta)) - H_2(\eta) \geq 0 \iff I_2(\xi,\eta) \leq H_2(\xi).$$

In this case, the equivalence $I_2(\xi, \eta) = H_2(\xi)$ holds if and only if ξ is a function of η. This proves the right-hand side of Equation 5.7 and Statement 4 of Theorem 5.3. □

B Algorithms

B.1 General Mutual Information I_2

Algorithm B.1 (General Mutual Information I_2)

```
function Ixy = gmi(X,Y)
begin
    /* Ranking of the time series */
    rX = ranking(X);
    rY = ranking(Y);
    rXY = cat(rX,rY);

    /* Iteration process to determine epsilon */
    epsilon = determine_epsilon(rXY);

    /* Compute entropy */
    hx = h(rX, epsilon);
    hy = h(rY, epsilon);
    hxy = h(rXY, epsilon);

    /* Compute normalized GMI */
    Ixy = (hx + hy - hxy) / hy;
end /* function */
```

This pseudo-code function describes the basic calculation steps to obtain the normalized General Mutual Information between the, possibly multi dimensional, time series X and Y. It brings together the previously mentioned theoretical fragments, like ranking and entropy computation, as described earlier in Chapter 5 The coarseness parameter `epsilon` is determined iteratively in such a way that the entropy `h(rXY)` is sufficiently close to a predefined constant.

Algorithm B.2 (Ranking)

```
function rX = ranking(X)
begin
    for n=1:NumberOfTimeSeries(X)

        /* Determine sort index */
        [sortX(n,:),sort_index_X(n,:)]=sort(X(n,:));

        /* Resort ascending sequence according to */
        /* the corresponding sort index.          */
        rX(n,sort_index_X(n,:))=...
          linspace(0.0,1.0,length(X(n,:)));
    end /* for n */
end /* function */
```

The term ranking denotes the transformation of an arbitrarily distributed time series into its uniformly distributed counterpart. The process of ranking is basically implemented as a sorting algorithm, from which only the resulting restructuring index is taken. The restructuring index is used for reordering an ascending sequence to reconstruct the shape of the original signal.
This function takes arbitrarily distributed time series $\{x_i\}$ and transforms them into sequences $\{r_i\}$ of uniformly distributed relative rank numbers. The relative rank numbers are distributed in the interval $[0, 1]$, as described in Section 5.3.2.

Algorithm B.3 (Entropy)

```
function hX = h(rX, epsilon)
begin
    lenVec=LengthOfTimeSeries(rX);
    numVec=NumberOfTimeSeries(rX);

    /* Compute the conjunction of all binary */
    /* rank distance matrices.               */
    for n=1:lenVec
        xBinVec = ...
          abs(rX(1,1:lenVec-n)-rX(1,1+n:lenVec))<epsilon;

        for i = 2:numVec
          xBinVec = xBinVec & ...
            (abs(rX(i,1:lenVec-n)-rX(i,1+n:lenVec))<epsilon);
        end /* for i */

        /* Successively update correlation integral. */
        Cx=Cx+sum(xBinVec);
    end /* for n */

    /* Finally, compute the entropy from the value */
    /* of the correlation integral Cx.             */
    hx = -log2((1.0/lenVec)*(1.0+((2.0/lenVec)*Cx)));
end /* function */
```

This function implements the concept of the binary rank distance matrices and its calculus, as described in Section 5.3.3. In order to minimize the memory requirement, this algorithm successively computes the correlation integral `Cx` on the basis of the secondary diagonals in the binary rank distance matrices.

B.2 Backward Elimination

Algorithm B.4 (Backward Elimination)

```
function removed_vars = BackwardElimination (X,Y)
begin
    removed_vars=[];
    acutal_X=X;

    for k=1:(NumberOfTimeSeries(X) - 1)
        /* Compute reference GMI for current iteration. */
        ref_gmi=gmi(actual_X,Y);

        tmp_gmi(v)=[];
        for v ∈ SetOfIndicies(actual_X)
            /* GMIs with temporarily removed variables. */
            tmp_gmi(v) = gmi(RemoveVar(actual_X, v),Y);
        end /* for v */

        /* Get variable with smallest information gain. */
        var_index = min(ref_gmi - tmp_gmi);

        /* Remove this variable.*/
        actual_X=RemoveVar(actual_X, var_index);
        removed_vars(k) = var_index;
    end /* for k */

    /* Append last remaining variable.*/
    removed_vars(k+1) = SetOfIndicies(actual_X);
end /* function */
```

The backward elimination strategy starts out with a complete set of input variables. It removes successively those input variable from `X` which show the least information gain between two iterations. Finally, backward elimination returns the sequence of removed variable indices. The last entry of this sequence depicts the variable with the highest information gain.

B.3 Forward Selection

Algorithm B.5 (Forward Selection)

```
function [vars,tlags]=ForwardSelection(X,Y,maxDim,maxTLags)
begin
    vars=[];
    tlags=[];

    /* Select the specified max. number of variables. */
    for k=1:maxDim
        tmp_X = SelectTimeSeries(X,vars);
        tmp_X = ApplyTimeLags(tmp_X,tlags);
        tmp_gmi = [];

        /* Scan along variables. */
        for d=1:NumberOfTimeSeries(X)
            X_d = SelectTimeSeries(X,d);

            /* Scan along time lags. */
            for t=0:maxTLags
                X_dt = ApplyTimeLags(X_d,t);

                /* GMI with temporarily added variable. */
                tmp_gmi(d,t)=gmi((tmp_X ∪ X_dt),Y);
            end /* for t */
        end /* for d */

        /* Get variable, time lag combination */
        /* with highest information gain.     */
        [var_index, t_lag] = max(tmp_gmi);

        /* Record the selected variable, time lag */
        /* combination.                           */
        vars(k) = var_index;
        tlags(k) = t_lag;
    end /* for k */
end /* function */
```

In contrast to the backward elimination strategy, forward selection starts out with an empty feature set. In each iteration, it successively adds one particular feature which provides the highest information gain among all

available alternatives. The fact that forward selection starts out with an empty feature set, renders the identification of implicit time dependencies feasible in the first place.

C Feature Selection Strategies with the GMI

C.1 Global Selection Strategy

In Tables C.1, C.2, C.3 and C.4, the indices of the input variable constellations with the highest GMI values among all possible input combinations are depicted. The set of input variables x in the expression $I_2(x, y_1)$ is defined as $x = \{x_{i_k} | i_k \neq 0,\ k = 1, ..., D\}$, where D is the cardinality of the feature subset, which contains the chosen variables from the N dimensional superset of all features. In this particular case, the cardinality of the feature subset, i.e. the number of variables to be chosen is $D = 7$.

	x_{i_1}	x_{i_2}	x_{i_3}	x_{i_4}	x_{i_5}	x_{i_6}	x_{i_7}	$I_2(x, y_1)$
1	1	0	0	0	0	0	0	0.1342
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
23259	3	4	6	7	9	10	11	0.6138
23260	3	4	6	7	9	10	12	0.6308
23261	**3**	**4**	**6**	**7**	**9**	**10**	**13**	**0.7333**
23262	3	4	6	7	9	10	14	0.6547
23263	3	4	6	7	9	10	15	0.6493
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
26332	10	11	12	13	14	15	16	0.3724

Table C.1: *Indices of the input variable constellation around the maximum GMI value $I_2(x_{glob}, y_1)$ among all possible input combinations.*

	x_{i_1}	x_{i_2}	x_{i_3}	x_{i_4}	x_{i_5}	x_{i_6}	x_{i_7}	$I_2(x, y_2)$
1	1	0	0	0	0	0	0	0.2783
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
25848	5	7	8	11	12	13	14	0.8758
25849	5	7	8	11	12	13	15	0.8730
25850	**5**	**7**	**8**	**11**	**12**	**13**	**16**	**0.8758**
25851	5	7	8	11	12	14	15	0.8331
25852	5	7	8	11	12	14	16	0.8468
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
26332	10	11	12	13	14	15	16	0.7121

Table C.2: *Indices of the input variable constellation around the maximum GMI value $I_2(x_{glob}, y_2)$ among all possible input combinations.*

	x_{i_1}	x_{i_2}	x_{i_3}	x_{i_4}	x_{i_5}	x_{i_6}	x_{i_7}	$I_2(x, y_3)$
1	1	0	0	0	0	0	0	0.3565
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
22202	2	5	7	8	9	13	14	0.9195
22203	2	5	7	8	9	13	15	0.9187
22204	**2**	**5**	**7**	**8**	**9**	**13**	**16**	**0.9221**
22205	2	5	7	8	9	14	15	0.8440
22206	2	5	7	8	9	14	16	0.8468
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
26332	10	11	12	13	14	15	16	0.7109

Table C.3: *Indices of the input variable constellation around the maximum GMI value $I_2(x_{glob}, y_3)$ among all possible input combinations.*

	x_{i_1}	x_{i_2}	x_{i_3}	x_{i_4}	x_{i_5}	x_{i_6}	x_{i_7}	$I_2(x, y_4)$
1	1	0	0	0	0	0	0	0.1600
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
22910	3	4	5	6	7	9	11	0.5238
22911	3	4	5	6	7	9	12	0.5193
22912	**3**	**4**	**5**	**6**	**7**	**9**	**13**	**0.7790**
22913	3	4	5	6	7	9	14	0.5884
22914	3	4	5	6	7	9	15	0.5821
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
26332	10	11	12	13	14	15	16	0.3023

Table C.4: *Indices of the input variable constellation around the maximum GMI value $I_2(x_{glob}, y_4)$ among all possible input combinations.*

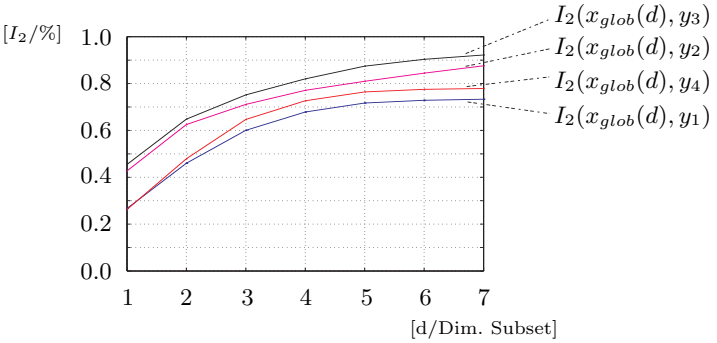


Figure C.1: *The maximum GMI $I_2(x_{glob}(d), y_j)$ as a function of the maximum number of selected variables in the feature subsets.*

Figure C.1 depicts the maximum GMI values as a function of the number of selected features for each output variable y_j. The set of input variables $x_{glob}(d)$ in Figure C.1 is defined as

$$x_{glob}(d) = \underset{v_i}{\operatorname{argmax}} \left[I_2(v_i, y_j) \right], \tag{C.1}$$

where $v_i = \{x_{i_k} \mid i_k \neq 0,\ k = 1, ..., d,\ i = 1, ..., G(N, d)\}$.

C.2 Backward Elimination Strategy

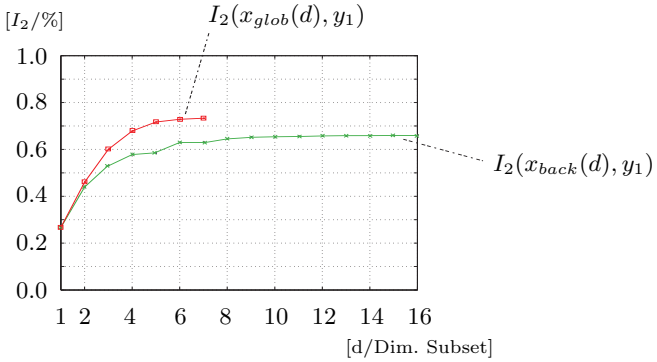


Figure C.2: *The GMI of the chosen variables from backward elimination* $I_2(x_{back}(d), y_1)$ *and the global maximum* $I_2(x_{glob}(d), y_1)$ *as a function of the number of selected variables in the feature subsets.*

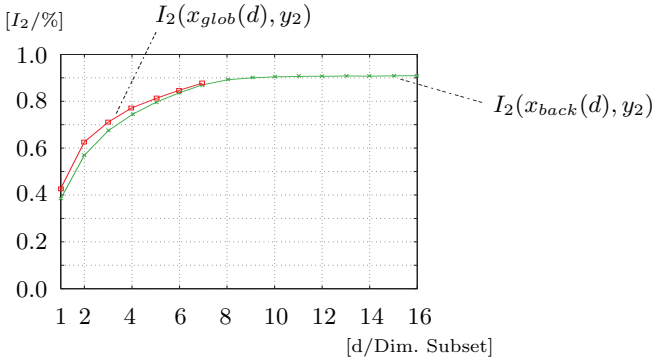


Figure C.3: *The GMI of the chosen variables from backward elimination* $I_2(x_{back}(d), y_2)$ *and the global maximum* $I_2(x_{glob}(d), y_2)$ *as a function of the number of selected variables in the feature subsets.*

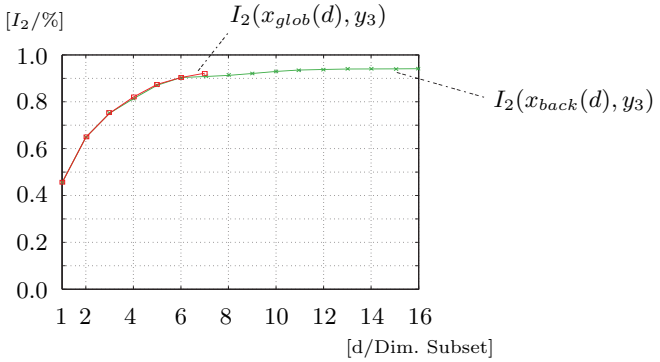


Figure C.4: *The GMI of the chosen variables from backward elimination* $I_2(x_{back}(d), y_3)$ *and the global maximum* $I_2(x_{glob}(d), y_3)$ *as a function of the number of selected variables in the feature subsets.*

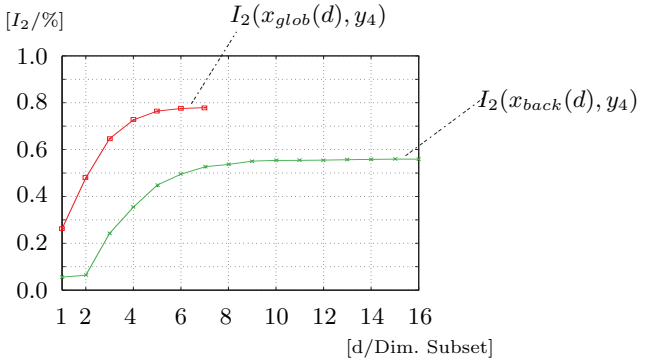


Figure C.5: *The GMI of the chosen variables from backward elimination* $I_2(x_{back}(d), y_4)$ *and the global maximum* $I_2(x_{glob}(d), y_4)$ *as a function of the number of selected variables in the feature subsets.*

The set of input variables $x_{back}(d)$ in the Figures above is defined for $d = N, ..., 2$ by the following iteration:

$$\begin{aligned} x_{back}(d) &= \{x_1, ..., x_d\}\,, \\ x_{back}(d-1) &= x_{back}(d) \setminus \underset{v_k}{\operatorname{argmin}} \left[I_2(x_{back}(d), y_j) - I_2(v_k, y_j)\right], \end{aligned} \tag{C.2}$$

where v_k is defined as $v_k = \{x_i \mid x_i \in x_{back}(d) \backslash \{x_{i_k}\},\ x_{i_k} \in x_{back}(d)\}$ and $k = 1, ..., d$.

C.3 Forward Selection Strategy

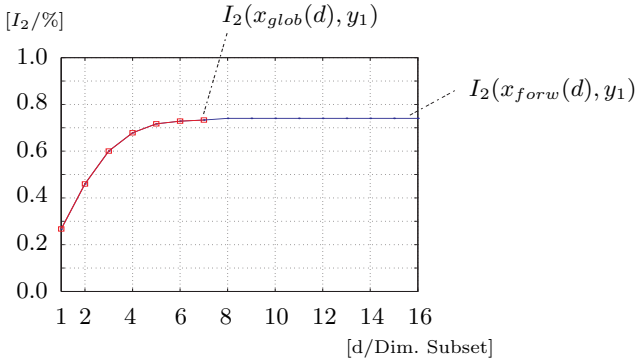


Figure C.6: *The GMI of the chosen variables from forward selection* $I_2(x_{forw}(d), y_1)$ *and the global maximum* $I_2(x_{glob}(d), y_1)$ *as a function of the number of selected variables in the feature subsets.*

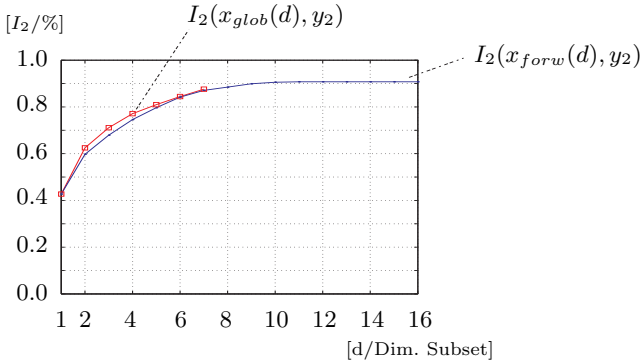


Figure C.7: *The GMI of the chosen variables from forward selection* $I_2(x_{forw}(d), y_2)$ *and the global maximum* $I_2(x_{glob}(d), y_2)$ *as a function of the number of selected variables in the feature subsets.*

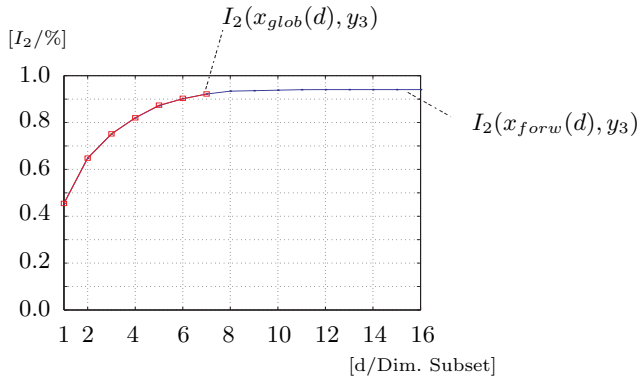


Figure C.8: *The GMI of the chosen variables from forward selection* $I_2(x_{forw}(d), y_3)$ *and the global maximum* $I_2(x_{glob}(d), y_3)$ *as a function of the number of selected variables in the feature subsets.*

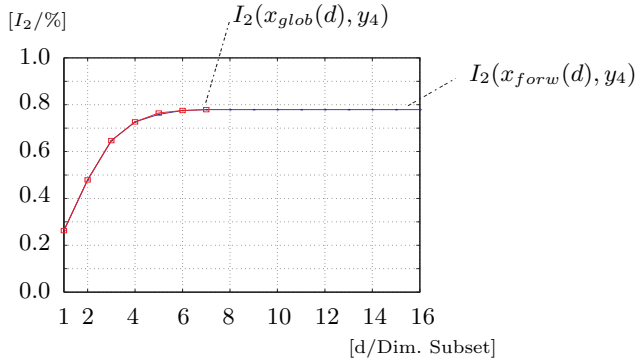


Figure C.9: *The GMI of the chosen variables from forward selection* $I_2(x_{forw}(d), y_4)$ *and the global maximum* $I_2(x_{glob}(d), y_4)$ *as a function of the number of selected variables in the feature subsets.*

The set of input variables $x_{forw}(d)$ in the Figures above is defined for $d = 1, .., N$ by the following iteration:

$$\begin{aligned} x_{forw}(0) &= \{\,\}\,, \\ x_{forw}(d) &= x_{forw}(d-1) \;\cup\; \underset{v_k}{\operatorname{argmax}}\,[I_2(v_k, y_j)]\,, \end{aligned} \tag{C.3}$$

where v_k is defined as $v_k = \{x_i \mid x_i \in x_{forw}(d-1) \cup \{x_k\}\}$. The subindex $k = 1, ..., N$ indicates the temporarily added variable.

D Application of the Forward Selection Strategy

D.1 Selection without the Consideration of Time Lags

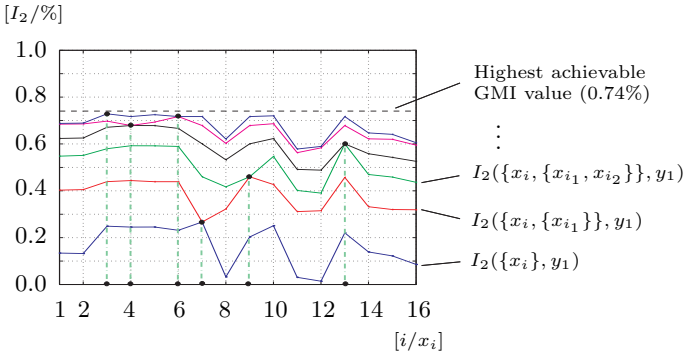


Figure D.1: *The GMI functions and the selected variables of all iteration cycles for output variable y_1. The variables comprising 99% of the highest achievable GMI value are: x_7, x_9, x_{13}, x_4, x_6, x_3.*

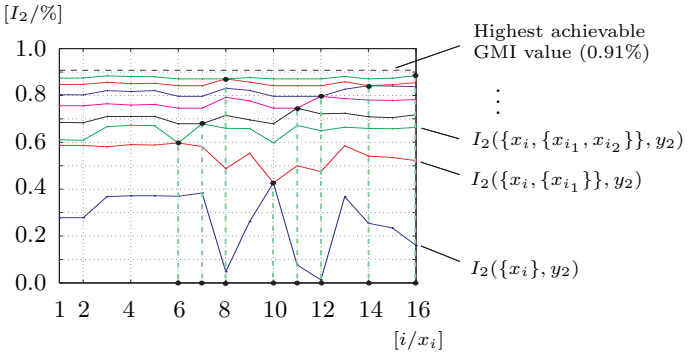


Figure D.2: *The GMI functions and the selected variables of all iteration cycles for output variable y_2. The variables comprising 99% of the highest achievable GMI value are: x_{10}, x_6, x_7, x_{11}, x_{12}, x_{14}, x_8,x_{16}.*

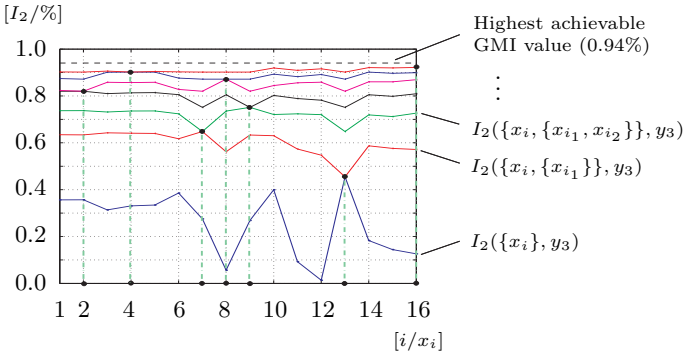


Figure D.3: *The GMI functions and the selected variables of all iteration cycles for output variable y_3. The variables comprising 99% of the highest achievable GMI value are: x_{13}, x_7, x_9, x_2, x_8, x_4,x_{16}.*

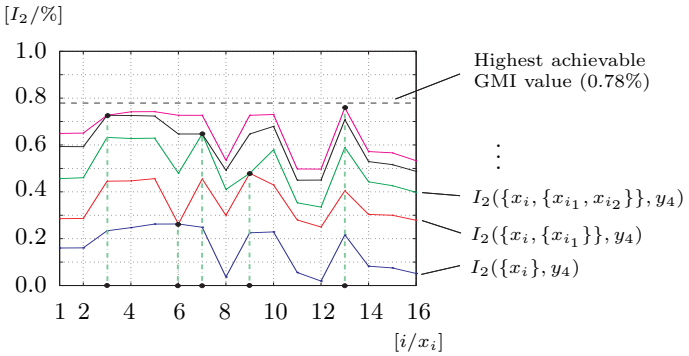


Figure D.4: *The GMI functions and the selected variables of all iteration cycles for output variable y_4. The variables comprising 99% of the highest achievable GMI value are: x_6, x_9, x_7, x_3, x_{13}.*

D.2 Selection including Time Lags

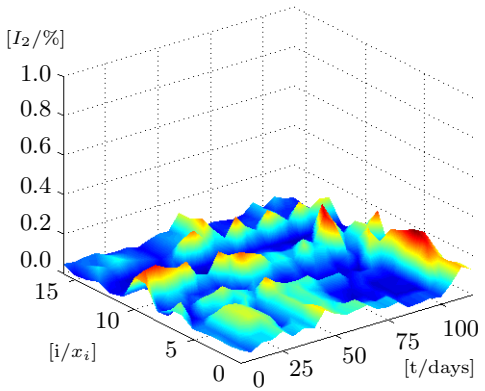


Figure D.5: *The GMI as a function of the input variable x_i and the time lag t for output variable y_1*

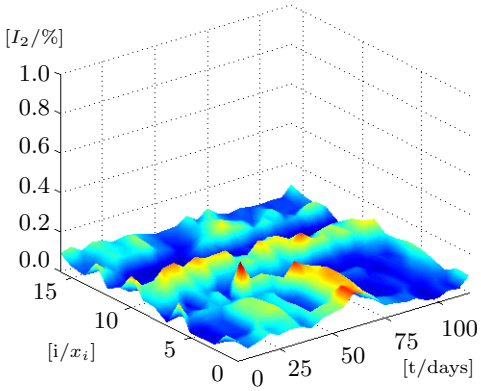


Figure D.6: *The GMI as a function of the input variable x_i and the time lag t for output variable y_2*

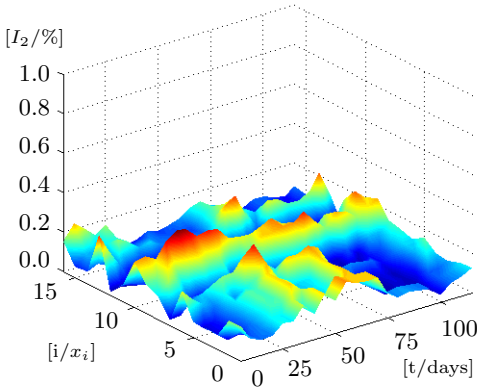


Figure D.7: *The GMI as a function of the input variable x_i and the time lag t for output variable y_3*

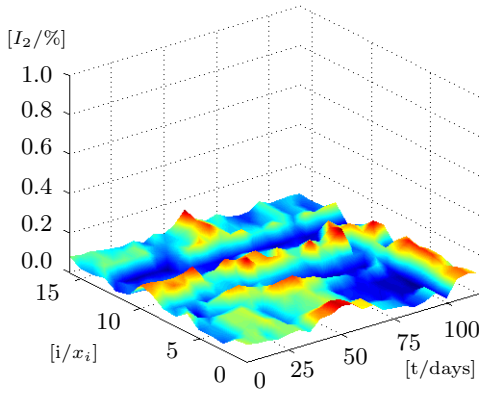


Figure D.8: *The GMI as a function of the input variable x_i and the time lag t for output variable y_4*

E Neural Identification of the Combustion Process

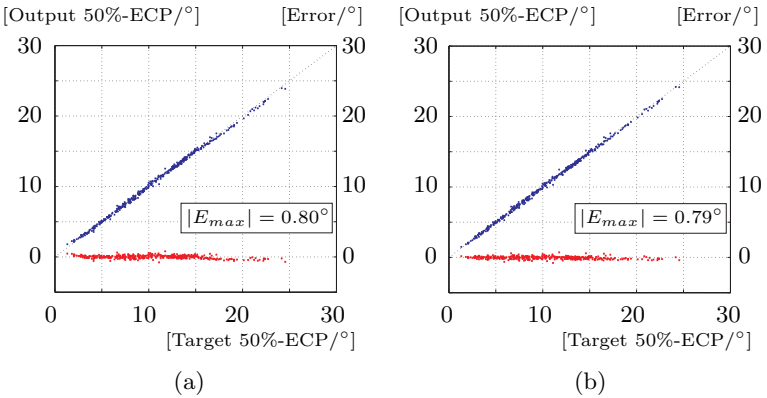


Figure E.1: *Neural identification results of the equidistantly sampled input variables for the determination of the 50% ECP. (a) Network performance after training with the RTRL algorithm, (b) Network performance after Extended Kalman Filter training.*

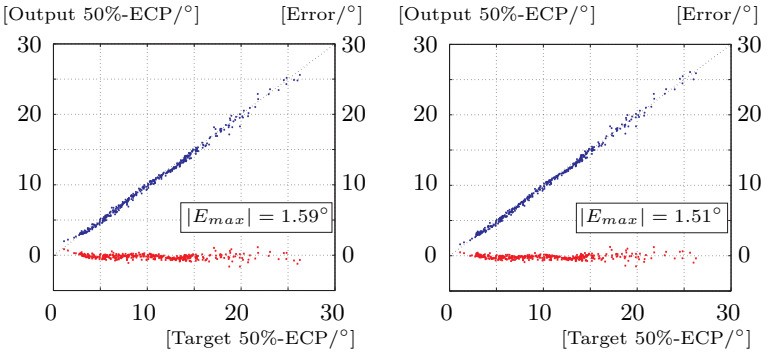


Figure E.2: *Neural identification results of the GMI selected input variables for the determination of the 50% ECP. (a) Network performance after training with the RTRL algorithm, (b) Network performance after Extended Kalman Filter training.*

F Curriculum Vitae

Persöhnliche Daten		
Geburtsort	22.12.1969	Hof, Bayern
Nationalität		deutsch
Familienstand		ledig
Schulen		
	1975-1980	Grund- und Hauptschule Kirchenlamitz / Weißenstadt
	1980-1989	Gymnasium Pegnitz
	1989	Abitur
Bundeswehr		
	1989-1990	EloKa / Fernmeldeaufklärung
Universität		
	10/1990-04/1997	Informatikstudium an der Universität Würzburg
	1993-1994	Akademisches Jahr an der University of Texas at Austin
	1996	Diplomarbeit in der Forschungsabteilung der Daimler-Benz AG

Diplomarbeit

Recurrent Multilayer Perceptrons for Identification and Control with Extended Kalman Filter Training

Wissenschaftliche Veröffentlichungen

F. Heister und R. Müller, An Approach for the Identification of Nonlinear, Dynamic Processes with Kalman Filter Trained Recurrent Neural Networks, Beiträge zur Leistungsbewertung verteilter Systeme der Universität Würzburg, 1999

F. Heister und G. Schock, Nonlinear, statistical Data-Analysis for the optimal Construction of Neural-Network Inputs with the Concep of a Mutual Information, ES2000-23, European Symposium on Artificial Neural Networks Proceedings, 2000

F. Heister und M. Fröhlich, Nonlinear Time Series Analysis of Combustion-Pressure Data for Neural Network Training with the Concept of a Mutual Information",
Proc IMechE, Vol 215, Part D, 2001

Beruflicher Hintergrund

08/1991	Werkstudent bei Siemens, Abt. Projektmanagement ICE
09/1992-10/1992	Dozent in der Berufsfortbildung der Bayerischen Arbeitgeber-verbände
11/1994-03/1995	Wissenschaftliche Kraft am Lehrstuhl für Informatik
05/1997-06/1997	Dozent in der Berufsfortbildung der Bayerischen Arbeitgeber-verbände
07/1997-06/2000	DaimlerChrysler AG, Esslingen
07/2000-04/2002	Sonem, GmbH, Würzburg
10/2002-heute	F.Hoffmann-La Roche AG, Schweiz

List of Figures

Bibliography

[Bar95a] M. Bargende. Schwerpunkt-Kriterium und automatische Klingelerkennung. *MTZ*, 56:632–638, 1995.

[Bar95b] M. Bargende. Verbrennungs- und Ladungswechselanalyse. In *Stuttgarter Symposium Kraftfahrwesen und Verbrennungsmotoren*, pages M8.1–M8.16, Stuttgart, 1995.

[BDSL91] W.A. Brock, W.D. Dechert, J.A. Scheinkman, and B. LeBaron. A test for independence based on the correlation dimension. Technical report, University of Wisconsin, 1991.

[Bel61] R Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[Bis99] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, 1999.

[Bos94] Bosch. *Autoelektrik, Autoelektronik am Ottomotor*. VDI Verlag, Stuttgart, 2nd edition, 1994.

[Car91] J.-F. Cardoso. Super-symmetric decomposition of the fourth-order cumulant tensor. blind identification of morde sources than sensors. In *ICASSP'91*, pages 3109–3112, 1991.

[CC87] C.K. Chui and G. Chen. *Kalman Filtering with Real-Time Applications*. Springer-Verlag, 1987.

[CC01] T.F. Cox and M.M.A. Cox. *Multidimensional Scaling*. Chapman and Hall, 2001.

[Chi61] A. J. Chintschin. Der Begriff der Entropie in der Wahrscheinlichkeitsrechnung. In *Arbeiten zur Informationstheorie I*, volume 2, pages 7–29. Deutscher Verlag der Wissenschaften, Berlin, 1961.

[CHL96] J.-F. Cardoso and B. Hvam Laheld. Equivariant adaptive source separation. *IEEE Trans. on Signal Processing*, 44(12):3017–3030, 1996.

[Com94] P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36:pp. 287–314, 1994.

[Csa81] P. Csallner. Eine Methode zur Vorausberechnung der Änderung des Brennverlaufs von Ottomotoren bei geänderten Betriebsbedingungen. Dissertation, Fakultät für Maschinenwesen, Technische Universität München, 1981.

[Fad56] D.K. Faddejew. Zum Begriff der Entropie eines endlichen Wahrscheinlichkeitsraumes. In *Arbeiten zur Informationstheorie I*, volume 2, pages 86–90. Deutscher Verlag der Wissenschaften, Berlin, 1956.

[Fod02] Imola K. Fodor. A survey of dimension reduction techniques. Technical Report, Lawrence Livermore National Laboratory, 2002.

[Fri94] Bernd Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[Frö03] M.H. Fröhlich. Informationstheoretische Optimierung künstlicher neuronaler Netze für den Einsatz in Steuergeräten. Dissertation, Fakultät für Informatik und Kognitionswissenschaften, Eberhard-Karls-Universität Tübingen, 2003.

[GSC91] P. Grassberger, T. Schreiber, and Schaffrath C. Nonlinear time sequence analysis. *Int. J. Bifurcation and Chaos*, 1:521–547, 1991.

[Har28] R.V. Hartley. Transmission of Information. *Bell Syst. Techn. Journal*, 7:pp. 535–563, 1928.

[HCO99] A. Hyvärinen, R. Cristescu, and E. Oja. A fast algorithm for estimating overcomplete ICA basis for image windows. *Proc. Int. Joint Conf. on Neural Networks*, pages 3917–3920, Washington D.C., 1999.

[Her94] W. Herden. A new combustion pressure sensor for advanced engine management. *SAE*, 940379:177–184, 1994.

[HF01] F. Heister and M. Froehlich. Non-linear time series analysis of combustion pressure data for neural network training with the concept of mutual information. *Journal of automobile engineering*, 215(D2):299–304, 2001.

[HM99] F. Heister and R. Müller. An approach for the identification of nonlinear, dynamic processes with Kalman-Filter-trained recurrent neural structures. Technical report, FT2/EA 0000-99, DaimlerChrysler AG, Abtl. FT2/EA, 70546 Stuttgart, 1999.

[HS00] F. Heister and G. Schock. Nonlinear, statistical Data-Analysis for the optimal Construction of Neural-Network Inputs with the Concept of a Mutual Information. In *ESANN'2000 proceedings - European Symposium on Artificial Neural Networks, Bruges (Begium)*, pages 439–444, 26-28 April 2000.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[Hyv97] A. Hyvärinen. A family of fixed-point algorithms for independent component analysis. *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Munich, Germany, 1997.

[Hyv98] A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing Systems*, volume 10, pages 273–279, MIT Press, 1998.

[Hyv99] A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:pp. 94–128, 1999.

[Jac91] J.E. Jackson. *A guide to Principle Components*. John Wiley and Sons, New York, 1991.

[JH91] C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:pp. 1–10, 1991.

[Jol86] I.T. Jolliffe. *Principal Component Analysis*. Springer, 1986.

[Kar00] J. Karhunen. Nonlinear independent component analysis. In *Independent Component Analysis: Principles and Practise*. Cambridge University Press, Cambridge, UK, 2000.

[KJ98] R. Kohavi and G. John. The wrapper approach. In *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Springer Verlag, 1998.

[KL51] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:76–86, 1951.

[Koh88] T. Kohonen. The neural phonetic typewriter. *IEEE Computer*, 21:11–22, 1988.

[Koh90] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78:1480–1484, 1990.

[KPO98] J. Karhunen, P. Pajunen, and E. Oja. The nonlinear pca in blind source separation: Relations with other approaches. *Neurocomputing*, 22:5–20, 1998.

[KS96] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[Kur94] R Kuratle. Meßzündkerzen mit integriertem Drucksensor. *Motortechnische Zeitschrift MTZ*, 55(2):120–126, 1994.

[Lam96] R.H. Lambert. *Multichannel Blind Deconvolution: FIR Matrix Algebra and Separation of Multipath Mixtures*. PhD thesis, Univ. of Southern California, 1996.

[Law97] Wolfhard Lawrenz. *CAN System Engineering - From Theory to Practical Applications*. Springer-Verlag, 1997.

[Lee01] T.-W. Lee. *Independent Component Analysis: Theory and Applications*. Kluwer Academic Publishers, 2001.

[Lof90] O. Loffeld. *Estimationstheorie*. Oldenbourg, Siegen, 1990.

[LS98a] M. Lewicki and T.J. Sejnowski. Inferring sparse, overcomplete image codes using an efficient coding framework. *Advances in Neural Information Processing*, 10:815–821, MIT Press, 1998.

[LS98b] M. Lewicki and T.J. Sejnowski. Learning overcomplete representations. *Advances in Neural Information Processing*, 10:556–562, MIT Press, 1998.

[LWB95] U. Lindgren, T. Wigren, and H. Broman. On local convergence of a class of blind separation algorithms. *IEEE Trans. on Signal Processing*, 43:3054–3058, 1995.

[MB99] W. Maass and C.M. Bishop. *Pulsed Neural Networks*. The MIT Press, Cambridge, MA, 1999.

[MKB95] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis, Probability and Mathematical Statistics*. Accademic Press, 1995.

[MP43] W.S. McCulloch and W.H. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activities. *Bull. Math. Biophys.*, Vol. 5:pp. 115–133, 1943.

[MP88] M. Minsky and S. Papert. Perceptrons. *Neurocomputing: Foundations of Research, MIT Press*, 13:pp. 161–170, 1988.

[NS96] R.S. Neville and T.J. Stonham. Adaptive Critic for Sigma-Pi Networks. *Neural Networks*, Vol. 9(4):pp. 603–625, 1996.

[Pea88] J Pearl. *Probabilitstic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, California, 1988.

[PF94] G.V. Puskorius and L.A. Feldkamp. Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1994.

[PFX00] J.C. Principe, J.W. Fisher, and D. Xu. Information theoretic learning. In S. Haykin, editor, *Unsupervised Adaptive Filtering*, pages 265–319. Wiley, New York, 2000.

[PGJ92] D.-T. Pham, P. Garrat, and C. Jutten. Separation of a mixture of independent sources through a maximum likelihood approach. In *EUSIPCO*, pages 771–774, 1992.

[PH95] B. Pompe and M. Heilfort. On the Concept of the Generalized Mutual Information Function and Efficient Algorithms for Calculating it. 1995.

[PKTS89] R. Pischinger, G. Kraßnig, G. Taucar, and Th. Sams. *Thermodynamik der Verbrennungskraftmaschine*. Springer, Wien, 1989.

[Pom93] B. Pompe. Measuring statistical dependencies in a time series. *J Stat. Phys.*, 73:587–610, 1993.

[Pom97] B. Pompe. *Die Messung von Informationsflüssen mit einer verallgemeinerten Transinformation*. Math.-Naturw. Fakultät, Ernst-Moritz-Arndt-Univ. Greifswald, Januar 1997.

[Pom98] Bernd Pompe. Ranking and Entropy Estimation in Nonlinear Time Series Analysis. Technical report, E.-M.-Arndt-Universität Greifswald, Institut für Physik, 1998.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J.R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, Los Altos, California, 1993.

[RB61] A. Rényi and J. Balatoni. Über den Begriff der Entropie. In *Arbeiten zur Informationstheorie I*, volume 2. Deutscher Verlag der Wissenschaften, Berlin, 1961.

[Rén77] A. Rényi. *Wahrscheinlichkeitsrechnung, mit einem Anhang über Informationstheorie*. Deutscher Verlag der Wissenschaften, Berlin, Berlin, 5nd edition, 1977.

[Rip96] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[RM86] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, volume 1: Foundations. MIT Press, Boston, MA, 1986.

[Ros60] F. Rosenblatt. Perceptron Simulation Experiments. *Proc. IRE.*, Vol. 48:pp. 301–309, 1960.

[Sch95] M. Schwervish. *Theory of Statistics*. Springer, 1995.

[Sea91] L. Schweizer et al. A fully neural approach to image compression. In T. Kohonen and O. Mäkisara, editors, *Artificial Neural Networks*, pages 815–820. North-Holland, Amsterdam, 1991.

[Sha48] Claude. E. Shannon. A mathematical theory of communcation. *The Bell System Technical Journal*, 2:379–423, 623–656, 1948.

[Spe04] C. Spearman. The Proof and Measurement of Association Between Two Things. *American Journal of Psychology*, 15:72–101, 1904.

[SPP06] I. Santamaria, P.P. Pokharel, and J.C. Principe. Generalized correlation function: Definition, properties, and application to blind equalization. *IEEE Transactions on Signal Processing*, 54(6):2187–2197, 2006.

[SW89] S. Singhal and L. Wu. Training Multilayer Perceptrons with the Extended Kalman Algorithm. In *Advances in Neural Information Processing Systems*, volume 1, pages 133–140. Morgan Kaufmann, 1989.

[Tak83] F. Takens. Invariants related to dimension and entropy. *Atas do 13. Col. brasiliero de Matematicas*, 1983.

[TJ91] A. Taleb and C. Jutten. Nonlinear source separation: The post-linear mixtures. volume 24, pages 21–29, 1991.

[WZ89] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[Zel94] A. Zell. *Simulation Neuronaler Netze*. Addison–Wesley, Bonn, 1994.

Index