Künstliche neuronale Netze in der Neurochirurgie

Diplomarbeit

gemäß § 20 der Prüfungsordnung vom 10. Mai 1989 des Fachbereiches
Angewandte Informatik und Mathematik
der Fachhochschule Fulda
Zugeordnetes Fach:
Künstliche neuronale Netze

von

Gerhard Röhrig
&
Torsten Schreiber

Referent
Herr Professor Dr. Oleg Taraszow
Fachhochschule Fulda

Koreferent
Herr Dr. Bernd Hölper
Städtisches Klinikum Fulda

24. Dezember 1998

Inhaltsverzeichnis:

1 Einleitung

Im Rahmen eines Kooperationsvertrages zwischen der Fachhochschule Fulda und der Klinik für Neurochirurgie des Städtischen Klinikums Fulda soll in dieser Arbeit die Möglichkeit der Prognoseabschätzung von Patienten mit Schädel-Hirn-Verletzungen untersucht und ggf. realisiert werden.

Bei den in dem Projekt integrierten Teilbereichen handelt es sich um:

- Datenerfassung
- Datenverarbeitung
- Prognoseberechnung
- Visualisierung

Für diese Studie relevante Bereiche ergeben sich aus der Datenerfassung, der Datenverarbeitung und der Prognoseberechnung.
Untersucht wurden 30 Patienten mit prolongierter Aufwachphase. Mit Hilfe der Computertomographie und der Magnetresonanztomographie wurden die Daten der intra- und extraparenchymale Läsionen ermittelt und in einer Datenbank gespeichert.
In den folgenden Kapiteln werden diese Daten analysiert, klassifiziert und die Ergebnisse, die unter Verwendung nicht kommerzieller Simulatoren entstanden, beurteilt.

In Kapitel 2 wird eine kurze Einführung in den Themenbereich „künstliche neuronale Netze“ dargestellt. Das folgende 3. Kapitel beschäftigt sich mit den unterschiedlichen Modellen neuronaler Netze.
Nachdem die Grundlagen mit diesen Kapiteln besprochen wurden, geht es im Kapitel 4, den „Softwaresimulatoren“, um die Forschung auf dem nicht kommerziellen Markt nach geeigneten Simulatoren, die auf den Plattformen von „Microsoft Windows 95/98/NT“ installierbar sind.
Anschließend werden die Resultate anhand ausgewählter Simulatoren erörtert. Die Beschreibungen befinden sich im Kapitel 5 „HavBpNet++“, im Kapitel 6 „Turbo Pascal-Files“ und im 7. Kapitel der Stuttgarter Simulator „SNNSv4.1 für Windows“.

Die folgenden Kapitel beschreiben die Daten und deren Analyse. So wird im Kapitel 8 das Projekt „Multimodales Neuromonitoring“ besprochen und in Kapitel 9 die Ursprungsdatenbank des Städtischen Klinikums sowie die durch die neue Struktur bedingten Änderungen beschrieben.
Die Resultate, die durch das Trainieren und Testen der konvertierten Daten mit dem Stuttgarter Simulator „SNNSv4.1“ erzielt wurden, werden im 10. Kapitel dieser Arbeit interpretiert.
Die Ergebnisse der Regressionsanalysen werden im Kapitel 11 bzgl. der möglichen Kombinationen und der resultierenden Korrelationskoeffizienten nach Pearson beurteilt. Zusätzlich wird die lineare Regressions-/Korrelationsanalyse nach Filtrierung auf die Referenzen „Art“ und „Lage“ durchgeführt.
Kapitel 12 beschreibt die Funktionen und die Benutzung des im Rahmen dieser Arbeit entwickelte Analyseprogramms „KNN 1.0“.
Das Schlusswort im 13. Kapitel zieht das Resümee über die in dieser Arbeit erzielten Ergebnisse, beschreibt die Tendenzen für zukünftige Projekte in diesem Sektor der neuronalen Netze und stellt die Probleme innerhalb der Untersuchung der Daten heraus.
Im letzten Kapitel 14 befindet sich der kommentierte Quellcode des in „Visual Basic 5.0“ entwickelten Tool „KNN 1.0“. Die Kommentarzeilen sind farbig kenntlich gemacht.

2 Künstliche neuronale Netze: Eine Einführung

2.1 Natürliche neuronale Netze

Bevor wir uns den künstliche neuronalen Netzen zuwenden, wollen wir uns zunächst die Natur und die damit zugrundeliegenden natürlichen Modelle anschauen.
Die grundlegenden Bausteine in biologischen Systemen sind die Neuronen. Ein Neuron ist eine kleine Zelle, die elektrochemische Reize von mehreren Quellen empfängt. Als Reaktion darauf erzeugt ein Neuron elektronische Impulse, die wiederum an andere Neuronen oder Effektorzellen übertragen werden.
Das menschliche Nervensystem besteht aus ca. 10^{10} bis 10^{12} Neuronen, die jeweils mehrere Informationen speichern können.
Das Gewicht des menschlichen Gehirns liegt im Durchschnitt bei 1,5 kg, somit ergibt sich ein Gewicht für das Neuron von $1{,}5 * 10^{-9}$kg.

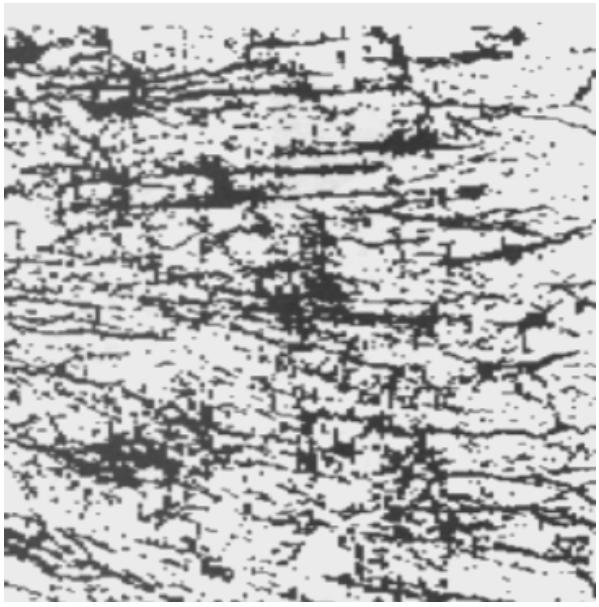


Abbildung 2.1 Die Hirnrinde

Neuronen sind komplexe Zellen, die auf elektrochemische Signale reagieren. Sie setzen sich zusammen aus einem Zellkern, einem Zellkörper, mehreren Dendriten, die über Synapsen „Eingabe-Verknüpfungen“ zu anderen Neuronen herstellen, sowie einem Axonstrang, der über Endkolben oder Synapsen ein AP (Aktionspotential) ausgibt.

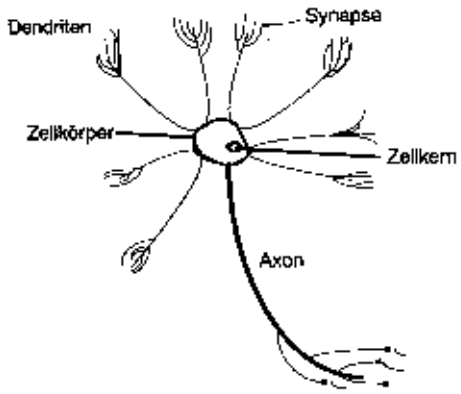


Abbildung 2.2 Das Neuron

Ein Neuron kann mit tausenden anderer Neuronen verknüpft sein, wobei die Verbindung über zwei Synapsentypen erfogt. Diese können erregender Natur (exzitatorisch) oder hemmend (inhibitorisch) sein.

Die neuronale Aktivität wird durch die Entstehung eines internen elektrischen Potentials, des sogenannten Membranpotentials, bestimmt. Das Potential wird durch die Eingabeaktivitäten anderer Zellen über die Synapsen je nach Typ verstärkt oder abgeschwächt. Wenn die aufsummierten, kumulativen Eingänge das Potential einer Zelle über einen Schwellenwert bringen, so feuert das Neuron, indem es eine Folge von Aktionspotentialen über das Axon ausschüttet, um andere Neuronen zu erregen oder zu hemmen. Die Frequenz der Impulsfortpflanzung reicht von 5 bis 125 kHz. Die Zeit, die ein Reiz zum Durchqueren einer Synapse benötigt, liegt bei ca. 1 ms. Nach einem Feuern braucht das Neuron eine Regenerationsphase von ca. 10 ms, während dieser Zeit ist das Neuron unempfindlich und kann nicht feuern. Die Aktivität eines Neurons wird über die Feuerfrequenz des von ihm erzeugten AP gemessen, die Spanne reicht hierbei von fünfzig bis einigen hundert Ausschüttungen pro Sekunde.

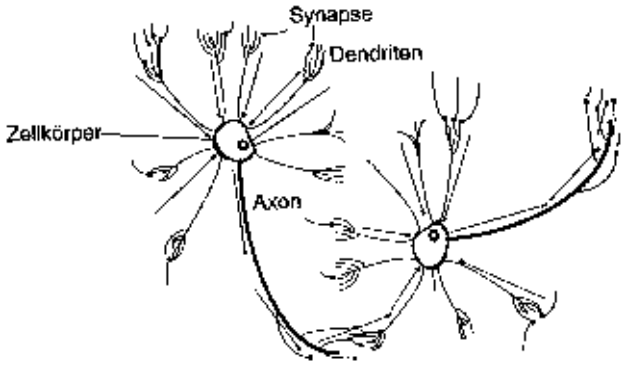


Abbildung 2.3 Signalübertragung zwischen Neuronen

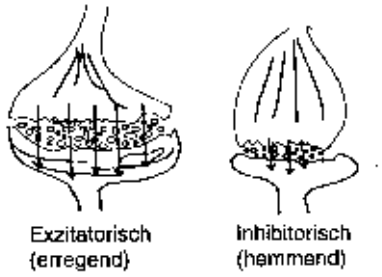


Abbildung 2.4 Allgemeine Synapsentypen

Bis heute ist nur wenig über den Lernprozess innerhalb des Gehirns bekannt. Man glaubt, dass innerhalb eines Neurons aufgrund erhöhter Zellaktivität eine Art metabolischen Wachstums entsteht, welches für das Lernen und das Gedächtnis verantwortlich ist. Dadurch wird die mögliche Ladung, die eine Synapse erzeugt, beeinflusst. Hier ist auf die Gewichte künstlicher neuronaler Netze, die wir später beschreiben, hinzuweisen.

Donald Hebb stellte als erster eine Behauptung auf, die auf den Lernprozess des Gehirns abzielte: „Wenn ein Axon der Zelle A nah genug an der Zelle B liegt, um diese zu erregen, und wiederholt oder andauernd feuert, erfolgt in einer der beiden Zellen ein Wachstumsprozess oder eine metabolische Veränderung, so dass sich A´s Einflusseffizienz auf B erhöht.“ Diese Aussage über das Lernen finden wir später bei den künstlichen neuronalen Netzen wieder.

2.2 Künstliche neuronale Netze

Um einen Einstieg in das Thema zu bekommen, beginnen wir mit einer einfachen Beschreibung für künstliche neuronale Netze :

Künstliche neuronale Netze sind Modelle der Gehirnfunktion. Sie versuchen, in Funktionsweise und Struktur Gehirnzellkomplexe nachzubilden und dadurch eine tragfähige Simulation komplexer menschlicher Denkvorgänge zu erzielen. Sie sind informationsverarbeitende Systeme, die sich aus primitiven, uniformen, miteinander kommunizierenden Verarbeitungseinheiten in großer Zahl zusammensetzen. Die dabei verarbeiteten und ausgetauschten Informationsquanten sind in fast allen Fällen unstrukturiert.

[frei nach K.P. Kratzer, Neuronale Netze, 2. Auflage]

Das erste abgeschlossene Modell im Bereich neuronaler Systeme stellte Frank Rosenblatt bereits im Jahre 1958 vor, das PERCEPTRON als Realisierung adaptiver klassifizierender Systeme. Trotz zahlreicher Modifikationen und Verbesserungen blieb der große Erfolg des Perceptrons aus.
Was folgte, war die von Widrow und Hoff 1960 entwickelte ADALINE (**ada**ptive **lin**ear **e**lement), ein adaptiver Schaltkreis mit Fehlerrückkopplung und eine Erweiterung dazu, der MADALINE-Komplex (**m**ultiple **ADALINE**).
In den 70er und frühen 80er Jahren wurde es ruhiger, die Forschungen zu diesem Thema wurden vorwiegend im militärischen Bereich sowie in der Neurophysiologie und der Kognitionswissenschaft fortgeführt. Beispielhaft seien hier Teuvo Kohonen und Steven Grossberg genannt. Kohonen befaßte sich mit der Selbstorganisation neuronaler Verarbeitungseinheiten und der Assoziationslehre. Grossberg untersuchte Lernverfahren, wie das Wettbewerbslernen (competetive learning). Seine Theorien wurden in späteren Netzmodellen aufgegriffen.
Im Jahre 1982 wurde von John Hopfield ein autoassoziatives Netz beschrieben, welches zur Abspeicherung und Rekonstruktion von Mustern geeignet ist und bis heute als Basismodell für Weiterentwicklungen dient.

Grundsätzlich werden heute vor- und rückwärtskoppelnde sowie konkurrierende Topologien (feedforward, backward, competetive) je nach Richtung der Informationsweitergabe während der Funktionsphase des künstlichen neuronalen Netzwerkes unterschieden. Durch Kombination können auch gemischte Topologien entstehen.

2.3 Historischer Überblick KNN

Als Übersicht wird nun ein chronologischer Abriß der Historie von künstlichen neuronalen Netzen gezeigt, die Bildung der ersten Modelle fällt in die Zeit , in welcher auch die ersten programmierbaren Computer entstanden.

1943 Erstes Modell zur Funktionsweise eines Neurons, entwickelt von W.S. McCulloch und W. Pitts

1949 Entwicklung der neurobiologische Grundlage des Lernens von Donald O. Hebb:
Hebbsche Regel: Die Verbindung zwischen zwei Neuronen wird verstärkt, sofern die post- und präsynaptische Zelle aktiv ist.

1958 Entwicklung des Perceptrons von F. Rosenblatt:
Perceptronkonvergenztheorem: „Existiert für ein gestelltes Problem eine Lösung als Perceptron, so führt sein Lernverfahren garantiert zum Erfolg.“

1960 Entwicklung des ADALINE (**ada**ptive **li**near **ne**uron) von B.Widrow und M.E. Hoff.
Delta - Regel

1969 Beweis der Notwendigkeit einer inneren neuronalen Schicht bei nicht linear separabler Problemstellung (Bsp.: XOR) durch M. Minsky und S. Papert

1972 Entwicklung eines Modells für den assoziativen Speicher durch T. Kohonen und J.A. Anderson („Korrelations-Matrix-Speicher“ bzw. „Neuronales Netz“)

Erweiterung der Hebbschen Regel: Die aktuelle Gewichtung einer Verbindung ergibt sich aus dem äußeren Produkt der Ein- / Ausgabeneuronen multipliziert mit einer Konstanten.

1974 Beschreibung des Lernalgorithmus Backpropagation durch P.J. Werbos. (siehe 1986 - Backpropagation)

1982 Entwicklung des Hopfield-Netzes durch J.J. Hopfield. Es stellt ein rückwärtsgekoppeltes Netzwerk mit Schwellenwerteinheiten und symmetrischen Verbindungen dar.

Ziel: Erreichung des Zustands minimaler Energie (Attraktorzustand) mit Hilfe der kollektiven Eigenschaften (emergent computational abilities).

1982 Entwicklung des Kohonen-Netzes durch T. Kohonen. Erstmaliges unüberwachtes Lernen wurde durch selbstorganisierender Bildung topologisch korrekter Merkmalskarten erreicht.

Ziel: Nach der Eingabe folgt die Aktivierung einer bestimmten Region. Ähnliche Eingabemuster sollen dabei benachbarte Neuronen erregen.

1983 Entwicklung der Theorie des verstärkten Lernens (reinforcement learning) von A.G. Barto, R.S. Suttan und C.W. Anderson.

Ziel: Ein Neuron übernimmt die Funktion eines adaptiven Kritikelements, wobei die Ausgabeneuronen als Steuerelemente fungieren.

1983 Entwicklung eines Lernalgorithmus zum Training der inneren Schichten eines neuronalen Netzes von S. Kirkpatrick, C.D.Gelatt Jr. und M.P. Vecchi.

Ziel: Simuliertes Ausglühen (simulated annealing):

Der Zustand der niedrigsten Energie (globales Minimum) entspricht der Struktur eines idealen Kristallgitters. Um diesen Zustand zu erreichen, werden zu Beginn hohe Energiepotentiale erzeugt, um somit dem globalen Minimum näherzukommen.

1985 Die Boltzmann-Maschine konnte als erste Maschine innere Neuronen trainieren, wodurch die Lösbarkeit linear nicht separabler Problemstellungen ermöglicht wurde.

1986 Entwicklung des Backpropagations durch D.E. Rumelhart, G.E. Huntan und R.J. Williams

Verallgemeinerung der Delta-Regel von mehrschichtigen Perceptrons durch kontinuierliche, differenzierbare Aktivierungsfunktion.

Ziel: Der Fehler der Ausgabeeinheit wird rückwärts durch das Netz propagiert, wodurch die Verbindungsgewichte angepaßt werden.

1986 Anwendungsbeispiel NETtalk von T.J. Sejnowski und C.R. Rosenberg, das mit Hilfe eines dreischichtigen neuronalen Netzes gelernt hat, englische Wörter vorzulesen.

Bis heute Entwicklung kommerzieller und nicht kommerzieller Produkte, Unterstützung der Forschung.

2.4 Eigenschaften künstlicher neuronaler Netze

Positive Eigenschaften

Robustheit:

Durch die sogenannte „Kollektivverantwortung“ der simulierten Neuronen für erzielte Ergebnisse führt der Versagensfall einzelner Neuronen schlimmstenfalls zu einer leicht verminderten Leistung des Netzes. In der Systemtheorie wird dieses Verhalten als *graceful degradation* (schrittweise Verminderung) bezeichnet. Durch diese spezielle Eigenschaft sind neuronale Netze vorwiegend geeignet für Aufgaben, die folgende Charakteristika aufweisen:

- assoziierend
- interpolativ
- klassifizierend
- beurteilend

Modellinhärente Parallelisierungsmöglichkeit:

Die Datenstrukturen und Ausführungsmodelle von neuronalen Netzen eignen sich gut für Mehrprozessorarchitekturen. Die herkömmlichen halbautomatischen in Koprozeduren aufgelösten Algorithmen stehen im Gegensatz zu den hier verwendeten *massiv - parallelen* Ausführungsmodellen. Im Idealfall sollte je einer Zelle des Ausführungsmodells auch ein Prozessor zugeordnet sein. Dies ist jedoch im Allgemeinfall nicht möglich, da der Bedarf an Prozessoren viel zu hoch wäre, bzw. die hohe Anzahl der benötigten Kommunikationsverbindungen zwischen den Prozessoren nicht zu realisieren ist.
Der Versuch einer technischen Lösung geht dahin, dass man Netzmodelle verwendet, die in ihrer Verbindungsstruktur eingeschränkt sind oder eine Prozessoranordnung wählt, deren Verbindungsstruktur Punkt-zu-Punkt Verbindungen in einem hochdimensionalen Raum realisiert.

Adaptivität (Lernfähigkeit):
Das Wissen eines neuronalen Netzes für einen Anwendungsbereich liegt in seiner Verbindungsstruktur. Durch Modifizierung dieser Struktur ist es möglich, die Einflußnahme des Erregungszustands jeder Zelle auf die mit ihr verbundene Zelle zu regulieren. Das Netz kann durch Angabe von Eingangsvektoren und der genauen Beschreibung des gewünschten Outputs im Probe- oder Trainingsbetrieb lernen. Die Konfiguration eines Netzes sollte in der Lage sein, die erlernten Trainingsbeispiele zu reproduzieren und für alle nicht genau trainierten Fälle durch *Assoziation* oder durch *Interpolation* angemessene Outputs liefern.

Verteilte Wissensrepräsentation:
Wie wir gesehen haben, liegt das Wissen eines künstlichen neuronalen Netzes in den einzelnen Gewichtungen verteilt gespeichert. Daraus ergibt sich der weitere positive Aspekt einer höheren Fehlertoleranz des Gesamtsystems gegenüber einzelnen Neuronen und Verbindungen.

Höhere Fehlertoleranz:

Wird beim Entwurf eines künstlichen neuronalen Netzes darauf geachtet, dass die Dimensionierung und die Codierung der zu lernenden Werte so gewählt werden Fehler abzufangen, gelingt es durch die verteilte Wissensrepräsentation einzelne Fehler zu beheben.

Assoziative Speicherung von Information:

Herkömmliche Rechnerarchitekturen speichern ihr Wissen adressbezogen, d.h. gleichartiges Wissen wird weit voneinander ohne gemeinsamen Bezug zueinander abgespeichert. Demgegenüber kann ein künstliches neuronales Netz die Informationen inhaltsbezogen, also assoziativ, speichern. Dadurch ist es leicht, mit einem künstlichen neuronalen Netz ähnliche Eingabemuster zu klassifizieren.

Default-Werte und spontane Generalisierung:

Da künstliche neuronale Netze oft automatisch Eingabemuster klassifizieren und somit Prototypen bilden, wird es durch diese Generalisierung und Bildung von Default-Werten ermöglicht unvollständige Eingabemuster den gebildeten Klassen zuzuordnen.

Aktive Repräsentation:

Ein weiterer Pluspunkt der künstlichen neuronalen Netze gegenüber den herkömmlichen Programmstrukturen liegt in der Repräsentation. Während herkömmliche Architekturen durch aktive Programmkomponenten auf die passive Repräsentation zurückgreifen müssen, realisieren die künstlichen neuronalen Netze die Repräsentation aktiv, weil das Wissen in den Verbindungsgewichten gleichzeitig an der Verarbeitung beteiligt ist.

Nach dieser Auflistung vieler positiver Aspekte müssen, um das Gleichgewicht zu wahren, nun auch einige negative Eigenschaften künstlicher neuronaler Netze betrachtet werden.

Negative Eigenschaften

Wissenserwerb nur durch Lernen:

Will man einem küstlichen neuronalen Netz bereits ein Basiswissen mitgeben, wie dies bei KI-Systemen (KI = Künstliche Intelligenz) der Fall sein kann, so ist das nur für wenige Anwendungen künstlicher neuronaler Netze möglich. Als Beispiel seien hier die Hopfield-Netze im Einsatz bei Optimierungsproblemen genannt, wo die Gewichte durch externe Algorithmen vorbelegt werden. Im Normalfall erfolgt der Wissenserwerb künstlicher neuronaler Netze nur durch Lernen.

Relativ hoher Zeitaufwand zum Lernen:

Alle gängigen Lernverfahren lernen sehr langsam, besonders wenn die künstlichen neuronalen Netze voll vernetzt sind und alle Gewichte einzeln bestimmt werden müssen. Optimiert man bekannte Verfahren, werden die Probleme nicht vollständig gelöst, sondern bestenfalls reduziert.

Keine Selbstanalyse:

Künstliche neuronale Netze sind nicht in der Lage Introspektionen durchführen, d.h. die Analyse des Eigenwissens oder der Ablauf von Problemlösungen sind nicht auf einfache Art zu realisieren, wie dies bei KI-Systemen der Fall ist.

Logische Schlussfolgerungen fast nicht realisierbar:

Logische Schlussfolgerungen, wie einfache Wenn-Dann-Beziehungen, sind mit künstlichen neuronalen Netzen nur sehr schwer zu erreichen, da durch die kombinatorische Explosion die benötigte Anzahl an Neuronen und Verbindungen ins schier unermessliche anwächst.

Nach der Gegenüberstellung der positiven und negativen Aspekte der künstlichen neuronalen Netze kann man abschließend folgendes Fazit anfügen.

Die Welt der künstlichen neuronalen Netze läßt zwar ein sehr breites Spektrum als Anwendungsgebiet offen, es ist jedoch festzuhalten, dass verschiedene Faktoren von einem künstlichen neuronalen Netz nicht zu erwarten sind. Zum einen scheitert die genaue Abbildung des menschlichen Gehirns bereits am Mengengerüst, die Modelle geben nur elementare Strukturen wieder, welche die Theorien untermauern. Eine der Theorien ist es zum Beispiel, dass man vermutet ddas Lernen basiere auf Gewichtsveränderungen in den Verbindungen zwischen den einzelnen Neuronen.

Zum anderen befreit ein künstliches neuronales Netz die Softwareentwickler nicht von der Programmierarbeit. Weiterhin ist es notwendig, die Eingangsdaten so zu bearbeiten, dass sie eindeutige Klassifizierungen zulassen. Eine Nachbearbeitung der Outputdaten ist ebenfalls sinnvoll.

Deshalb muss eine sinnvolle Koexistenz von Programmierung, Datenbankmanagement und dem weiten Gebiet der künstlichen Intelligenz zugrunde gelegt sein, wobei für eine reibungslose Zusammenarbeit die Schnittstellen eindeutig gehalten werden und ein geeignetes Modell gefunden werden muss.

Man spricht bei diesen Modellen, die ein Wissensgebiet in mathematische, bzw. informationstechnische Systematiken transferieren, von sogenannten *Referenzmodellen*. Die Referenzmodelle haben die Aufgabe, sowohl die Terminologie als auch die Systemarchitektur zu normieren. Wir werden im Folgenden versuchen, die Terminologien und Formalismen im Bereich der künstlichen neuronale Netze darzustellen.

3 Modelle künstlicher neuronaler Netze

Zu Beginn ist es nötig, die vorherrschende Arbeitsweise des Neurons in das Referenzmodell zu transferieren. Das Wesentliche hierbei ist, dass man den Verlauf der Information, den Informationsfluß, beobachten, bzw. verschiedenen Verarbeitungseinheiten gesondert darstellen muss. Die Verarbeitungseinheiten untergliedern sich wie folgt:

- Die *Eingabeeinheiten* dienen als Eingabepuffer für von außen hereinkommende oder auch intern berechnete Informationen, d.h. klassifizierte Werte. Die Initialisierung der Eingabeeinheiten wird von außen vorgenommen und unterliegen nicht den internen Regeln des künstlichen neuronalen Netzes. Die Propagierung der Eingabewerte und die eingesetzten Lernregeln spiegeln den Momentanzustand des Netzes wider.

- Die *Ausgabeeinheiten* bilden die Schnittstelle des künstlichen neuronalen Netzwerkes nach außen.
  Die Interpretation unterliegt dem Menschen oder einem externen System. Zwischen den erwähnten Ein- und Ausgabeeinheiten liegen die

- *Zwischeneineinheiten*, die auch als *hidden units* bezeichnet werden, unterliegen einer netzinternen Verwaltung und sind ausschließlich vom Informationsfluß abhängig.

Einflussgrößen und Regeln bilden die Struktur dieser Verarbeitungseinheiten und damit auch des Netzes. Im folgenden werden diese genannt und danach explizit erklärt.

1. *Aktivierungszustand* *(state of activation)*
2. *Aktivierungsregel* *(activation rule)*
3. *Ausgaberegel* *(output rule)*
4. *Netzaktivität* *(net activity)*
5. *Fehlermaß* *(error quantifier)*

3.1 Der Aktivierungszustand

Eine Verarbeitungseinheit stellt in einem Gesamtsystem immer einen Teilaspekt zum Zeitpunkt t dar. Ist die Verarbeitungseinheit zu diesem Zeitpunkt t aktiviert, so kann man dies mit $a_i(t)$ beschreiben. Fasst man alle Verarbeitungseinheiten zusammen, so entsteht für diesen Zeitpunkt ein Vektor, der den Systemzustand durch ein bestimmtes Muster von Aktivierungen beschreibt. Somit kann sich eine sehr große Anzahl verschiedener Aktivierungen ergeben.

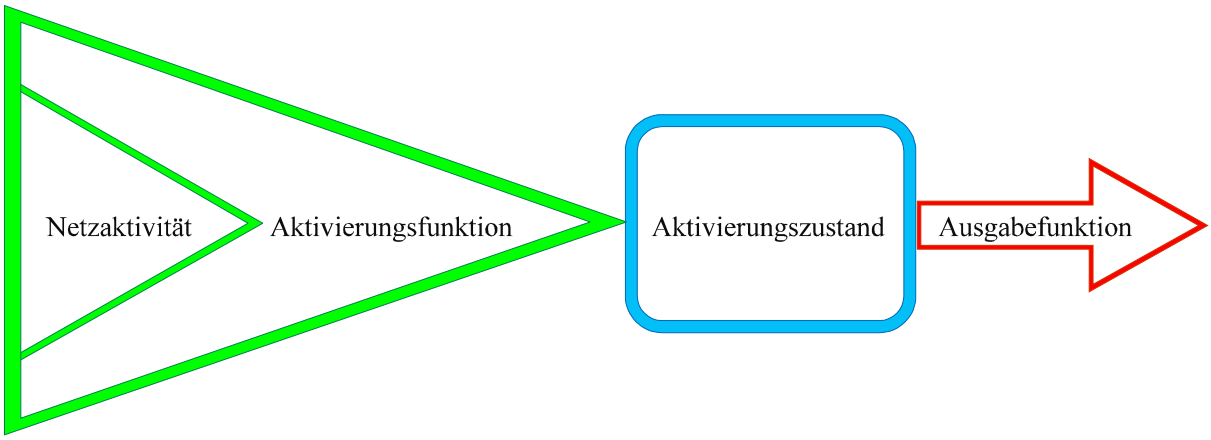


Abbildung 3.1 Informationsfluß in einer Verarbeitungseinheit

Die Aktivierungszustände können verschiedene Ausprägungen haben, einerseits können sie einem kontinuierlichem Wertebereich entnommen sein, andererseits können sie diskret sein.

Im diskreten Fall unterscheidet man zwischen binären Einheiten { 0,1 } für inaktiv/aktiv oder falsch/wahr und nicht binären Einheiten mit der Wertemenge { -1, 0, +1 } für falsch/indifferent/wahr.

Der kontinuierliche Fall bedient sich der Werte aus einem geschlossenen Intervall zwischen 0 und 1 ausschließlich der Werte 0 und 1.

Der Vorteil von künstlichen neuronalen Netzen mit diskreten Aktivierungszuständen liegt in ihrer schnelleren Adaption gegenüber der Netze mit kontinuierlichen Aktivierungszuständen .

Ist eine hohe Klassifikationsfähigkeit des Netzes gefordert, so ist es von Vorteil, wenn man auf Aktivierungszustände zurückgreift, die einem intervallförmigen Wertebereich entstammen.

3.2 Die Aktivierungsregel

Die Aktivierungen der Verarbeitungseinheiten richten sich an Werten, die an ihrem Eingang anliegen. Bei den Eingabeeinheiten sind das meist externe Werte, bei allen anderen Verarbeitungseinheiten, wie Zwischeneinheiten und Ausgabeeinheiten, sind es die von vorangestellten Einheiten gelieferten Werte, sofern sie einen Schwellenwert überschreiten.
Betrachtet man eine einzelne Einheit aus dem Gesamtsystem, so kann ihr Zustand als Netzaktivität in Bezug auf diese Einheit angesehen werden.

Die Netzaktivität an der Verarbeitungseinheit i zum Zeitpunkt t wird mit $net_i(t)$ bezeichnet.

Der neue Aktivierungszustand ergibt sich unter der Berücksichtigung der soeben dargestellten Netzaktivität $net_i(t)$ und dem Vorzustand der Verarbeitungseinheit:

$$a_i(t) = f_{akt}(net_i(t), a_i(t-1))$$

Die Aktivierungsfunktion f_{akt} kann verschiedene Formen haben, wie die Abbildungsmöglichkeit als *Identität* oder *lineare Abbildung*. Dabei wird die Netzaktivität entweder direkt oder über eine lineare Abbildung in den Aktivierungszustand übergeführt.
Vorteil dieser Aktivierungsfunktion ist die Einfachheit, als nachteilig erweist sich aber, dass das Klassifikationsvermögen stark eingeschränkt ist.
Die am meisten genutzte Klasse von Aktivierungsfunktionen ist die *Schwellwert-* oder *Treppenfunktion*.
Die Verarbeitungseinheiten können mit diskreten Aktivierungszuständen aktiviert werden.
Im simplen Fall läßt sich die Netzaktivität durch einen Schwellenwert modifizieren und das Gesamtergebnis wird mit einem konstanten Wert (z.B. 0) verglichen.

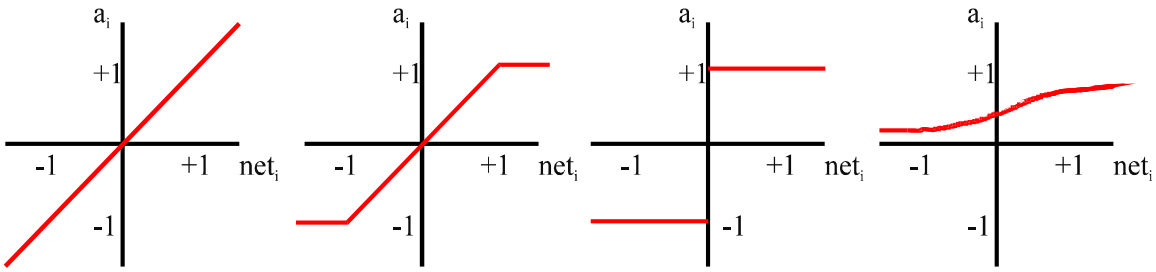


Abbildung 3.2 Lineare, Treppen- und Sigmoidfunktion

Andere Netze benötigen semilineare Aktivierungsfunktionen, wie die Sigmoidfunktion, die ein asymptotisches Verhalten in der Unendlichkeit zeigt und dabei gleichzeitig Grenzwerte für den Aktivierungszustand mit sich bringt.
Eine weitere Klasse für die Aktivierungsfunktion ist die *stochastische Aktivierung*. Sie geht von einer Aktivierungwahrscheinlichkeit *(firing rate)* aus. Hier liegt eine Interpretation innerhalb der Aktivierungsregel zugrunde, welche dann bestimmt, ob der alte Zustand beibehalten oder die Netzaktivität neu berechnet wird. Bei passender Größe des Parameters ist eine höhere Robustheit im Netz zu erzielen.

3.3 Die Ausgaberegel

Anhand des ermittelten Aktivierungszustandes in einer Verarbeitungseinheit i (Eingabeeinheit oder Zwischeneinheit) zu einem Zeitpunkt t kann die Weitergabe an die am Ausgang anliegende Verbindung als Ausgaberegel $o_i(t)$ definiert werden. Die Regel wird dabei beeinflusst von der Ausgabefunktion:

$$o_i(t) = f_{out}(a_j(t))$$

Am weitesten verbreitet innerhalb der verschiedenen künstlichen neuronalen Netzwerke ist, dass die Information ohne Veränderung weitergegeben wird - man spricht von der *Identität* als Ausgabefunktion.

Eine andere Möglichkeit besteht darin, dass die Ausgabefunktion Transformationen wie beispielsweise einen Normalisierungsschritt vornimmt und damit die Ausgaberegel bestimmt.

Als mathematische Beispiele werden anschließend die Treppen- oder Schrittfunktionen und die Sigmoide (S-förmige) Funktionen dargestellt.

Die Treppenfunktion läßt sich wie folgt definieren:

$$o_i = \begin{cases} 1, \text{ falls } net_i >= 0 \\ 0, \text{ sonst} \end{cases}$$

Eine Sigmoide Funktion hat im Schwellenwert höchste Sensibilität.
Als Beispiel können hier die logistische Funktion und auch der Tangens hyperbolicus aufgezeigt werden.

Für die *Logistische Funktion* gilt:

$$f(x) = 1 / (1 + e^{-x})$$

Die Ableitung stellt sich folgendermaßen dar:

$$\begin{aligned} f'(x) &= e^{-x} / (1 + e^{-x})^2 \\ &= [1 / (1 + e^{-x})] \cdot [e^{-x} / (1 + e^{-x})] \\ &= f(x) \cdot [(1 + e^{-x} - 1) / (1 + e^{-x})] \\ &= f(x) \cdot [1 - f(x)] \end{aligned}$$

Der *Tangens hyperbolicus* ist wie folgt definiert :

$$\begin{aligned} f(x) &= \tanh(x) \\ f(x) &= [(e^{x} - e^{-x}) / (e^{x} + e^{-x})] \\ f(x) &= 2 \cdot f_{log}(2x) - 1 \end{aligned}$$

Die Ableitung hierzu ist:

$$f'(x) = 1 - \tanh^2(x)$$

3.4 Das Fehlermaß

Das Fehlermaß bildet die Abweichung des erwarteten Verhaltens der Verarbeitungseinheit von der Soll-Lösung ab. Es ist durch einen Soll/Ist-Vergleich leicht zu ermitteln und bildet die Grundlage für Modifikationen der Verbindungsstruktur. Bei Eingabeeinheiten ist das Fehlermaß undefiniert, da die Aktivierung von außen vorgenommen wird.

3.5 Das Netzwerk

Da ein Netzwerk nicht nur aus einer Verarbeitungseinheit besteht, sondern aus vielen zusammengeschlossenen Einheiten, ist es notwendig, noch einige Begriffe zu klären. Diese sind:

- *Verbindungsstruktur*
- *Kontrollstrategie*
- *Vermittlungsregel*
- *Adaptionsregel*
- *Systemumgebung*

3.6 Die Verbindungsstruktur

Die Verbindungsstruktur zwischen den Verarbeitungseinheiten - sie sind die informationsverarbeitenden Instanzen innerhalb des Netzwerkes - muss die Verarbeitung des Aktivierungsmusters und den daraus resultierenden Informationsfluß gewährleisten. Gleichartige Zellen, die durch gleichartiges Ein-/Ausgabeverhalten und identische Rollen im Rahmen des Gesamtkommunikationsflusses gekennzeichnet sind, werden zu ***Schichten (Layern)*** zusammengefasst. Dies gilt auch für die Eingabe- und Ausgabezellen.

Die Verbindungsstruktur zwischen den Verarbeitungseinheiten trägt das Wissen eines neuronalen Netzes. Eine einzelne Verbindung zwischen einer Verarbeitungseinheit i und einer Verarbeitungseinheit j wird mit Hilfe von Gewichtungen gesteuert. Die gebräuchlichste Notation ist:

Gewicht der Verbindung von i nach j : w_{ij}

Die Art des Gewichtes wird durch den Betrag der Gewichtung dargestellt.
Erregende Gewichte (excitatory) haben eine positive Gewichtung und hemmende Gewichte (inhibitory) negative Gewichtungen.
Die Bedeutung einer Einzelzelle für den Gesamtkomplex ist charakterisiert durch den Betrag der eingehenden und ausgehenden Gewichte, man spricht vom „fan-in“ oder dementsprechend vom „fan-out“ . Die eingehenden Gewichte entstammen dem rezeptiven Feld einer Verarbeitungseinheit. Dies sind Folgezellen vorgeschalteter verbundener Neuronen, die in einem topologischen Zusammenhang stehen. Die folgende Abbildung 3.3 veranschaulicht die Begriffe.

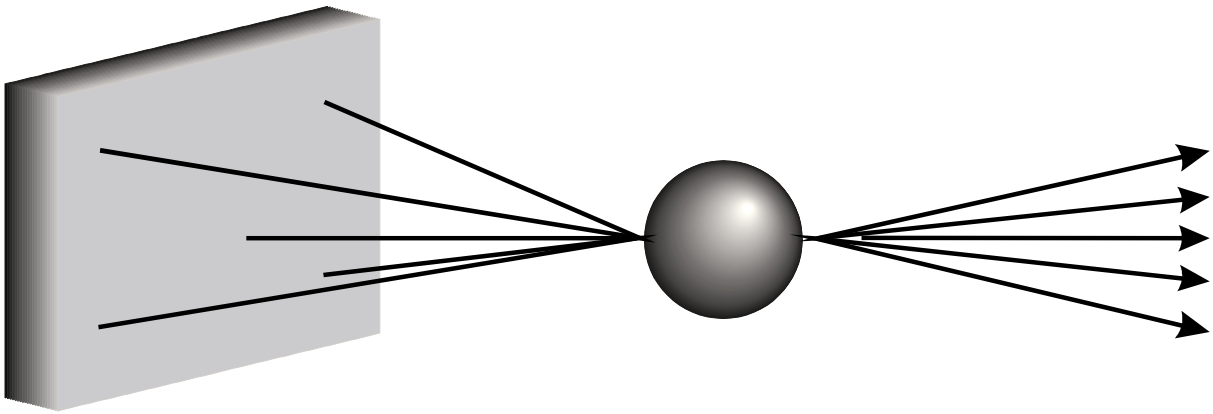


Abbildung 3.3 Rezeptives Feld, Fan-in, Verarbeitungseinheit, Fan-out

Es muss bei den Verbindungen unterschieden werden zwischen:

- **gerichteten Verbindungen** mit definierter Ausrichtung des Informationsflusses und
- **ungerichteten Verbindungen**, wobei die verbundenen neuronalen Zellen sich wechselseitig beeinflussen.

Sonderfälle der Gewichtungen sind die Schwellwerte (Bias), welche ein Maß für die Tendenz einer Verarbeitungseinheit i zur Aktivierung, bzw. zur Deaktivierung aufzeigen.

Schwellwert : θ_i

Künstliche neuronale Netze können verschiedene Topologien zeigen, so gibt es zum einen geschichtete Netze und zum anderen vollvernetzte Strukturen mit einer großen Anzahl an Variationen.

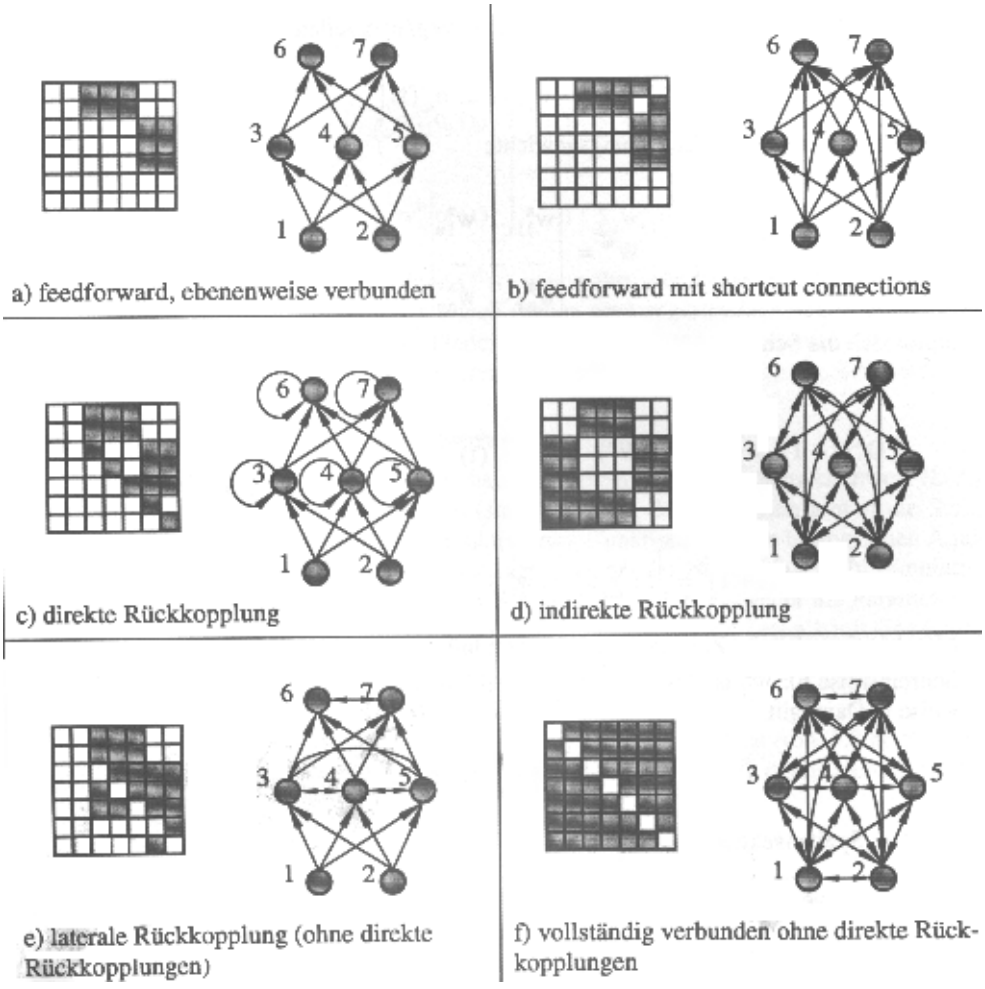


Abbildung 3.4 Beispiel-Topologien und ihre Verbindungsmatrizen

3.7 Die Kontrollstrategie

Unter der Kontrollstrategie versteht man die Aktivierungssequenz der Verarbeitungseinheiten. Sie ist nicht mit der Sequentialisierung bei Simulationen künstlicher neuronaler Netze in Ein-Prozessorsystemen zu verwechseln.
Prinzipiell kann man unterscheiden:

Vorwärtsvermittlung (feedforward propagation)
Vollvernetzte Strukturen
Mischformen

Bei der Vorwärtsvermittlung erfolgt die Aktivierung eines Netzes schichtweise von der Eingabeschicht bis zur Ausgabeschicht.
Vollvernetzte Strukturen erfordern eine Quasi–Gleichzeitigkeit der Aktivierung, die Kontrollstrategie reguliert die Verhaltensparameter.
Mischformen, wie zum Beispiel zunächst feedforward und dann voll vernetzt, erfordern eine selektive Kontrollstrategie. Diese selektive Kontrollstrategie zeigt sich besonders bei dem *„winner–take–all“* Konzept. Hier haben nur bestimmte Zellen mit bestimmter Aktivierungseigenschaft, die sie von anderen Zellen unterscheiden, die Möglichkeit, ihre Aktivierung an umgebende Zellen weiterzugeben.

3.8 Die Vermittlungsregel

Liegt eine Verbindungsstruktur vor und die Kontrollstrategie bestimmt, dass eine Zelle j zu aktivieren ist, wird durch die *Vermittlungsregel* aus den Aktivierungen aller vorgeschalteter Zellen sowie den Gewichten der Verbindungen die **Netzaktivität net_j** berechnet, die zur Aktivierung der Verarbeitungseinheit j dient.

$$net_j = \Sigma_i \; a_i \cdot w_{ij}$$

3.9 Die Adaptionsregel

Die Adaption des Anwendungswissens kann folgende Punkte umfassen:

- Änderung der Gewichte bestehender Verbindungen
- Auf- und Abbau von Verbindungen (Spezialfall von Gewichtsänderung, da w=0 einen logischen Abbau einer Verbindung bedeutet, theoretisch bleibt die Verbindung erhalten.)
- Änderung der Netztopologie

Die Gewichte werden durch einfache Algorithmen des Netzes in seiner Anwendungsumgebung adaptiert. In fast allen Fällen beruht die Adaption auf dem ermittelten Fehlermaß. Steht kein direkter Soll/Ist – Vergleich zur Verfügung, kann durch eine *Adaptions-Kontroll-Strategie* ein geeignetes Maß eingeführt werden. Ein Beispiel hierfür ist die *Fehlerrückvermittlung (back propagation).*

Man unterscheidet zwischen

- unüberwachtem Lernen (unsupervised learning)
- überwachtem Lernen (supervised learning)

Im Fall überwachten Lernens muss weiterhin unterschieden werden zwischen dem sogenannten

- Ein-/Ausgabelernen (hard learning), hier werden nur Eingabe- und Ausgabemuster präsentiert, und dem
- vollständigen Lernen (easy learning), Zwischenzell-Aktivierungen müssen angegeben werden.

Allen Adaptionsmethoden gemein ist die Parametrisierbarkeit, die die Stärke der Auswirkung einer Fehlleistung des Netzes auf die Gewichtskonfiguration beeinflusst. Der wichtigste Faktor ist der ***Lernfaktor*** η.

Exemplarisch werden an dieser Stelle die drei gebräuchlichsten Lernregeln dargestellt.

Hebbsche Regel

Wenn Zelle j eine Eingabe von Zelle i erhält und beide gleichzeitig aktiviert sind, dann erhöhe das Gewicht w_{ij}.

Der mathematische Zusammenhang kann wie folgt aufgezeigt werden :

$$\Delta w_{ij} = \eta \cdot o_i \cdot a_j$$

oder in allgemeiner Form

$$\Delta w_{ij} = \eta \cdot k(o_i, w_{ij}) \cdot g(a_j, t_j)$$

von – Funktion *nach - Funktion*

Legende	
Δw_{ij}	Änderung des Gewichtes w_{ij}
η	Lernfaktor (konstant)
o_i	Ausgabe der Vorgängerzelle
a_j	Aktivierung der Nachfolgezelle
t_j	Teaching input (erwartete Aktivierung)
$k(o_i, w_{ij})$	Funktion der Ausgabe und Gewicht der Vorgängerzelle
$g(a_i, t_j)$	Funktion der tatsächlichen und erwarteten Aktivierung

Tabelle 3.1 Legende der Hebb'schen Lernregeln

Delta-Regel (Widrow–Hoff)

Die Gewichtsänderung ist proportional zur Differenz (δ_j) der aktuellen Aktivierung (Ausgabe) und der erwarteten Aktivierung (Ausgabe).

Der mathematische Zusammenhang stellt sich hier wie folgt dar:

$$\Delta w_{ij} = \eta \cdot o_i (t_j - a_i) = \eta \cdot o_i \cdot \delta_j$$

$$\Delta w_{ij} = \eta \cdot o_i (t_j - o_i) = \eta \cdot o_i \cdot \delta_j$$

Legende	
Δw_{ij}	Änderung des Gewichtes w_{ij}
η	Lernfaktor (konstant)
o_i	Ausgabe der Vorgängerzelle
a_j	Aktivierung der Nachfolgezelle
t_j	Teaching input (erwartete Aktivierung)
δ_i	Differenz von akt. und erw. Ausgabe

Tabelle 3.2 Legende der Delta-Regel

Backpropagation Regel

Die Backpropagation Regel ist eine Erweiterung der Delta-Regel, wobei die Aktivierungsfunktion semilinear, d.h. sowohl monoton als auch differenzierbar sein muss.
Die Knotennummern müssen topologisch sortiert vorliegen.

$$\Delta w_{ij} = \eta \cdot o_i \cdot \delta_j$$

$$\delta_i = \begin{cases} f'_j(net_j) \cdot (t_j - o_i) & j \text{ ist Ausgabezelle} \\ f'_j(net_j) \cdot \Sigma_k \cdot (\delta_k \cdot w_{jk}) & j \text{ ist verdeckte Zelle} \end{cases}$$

Legende	
k	Summationsindex über alle direkten Nachfolgezellen

Tabelle 3.3 Legende der Backpropagation Regel

Beispiel :

$$net_j(t) = \Sigma_i o_i(t) \cdot w_{ij}$$

$f'_j = 1 / (1 + e\text{-}x)$ logistische Funktion

$o_j(t) = net_j(t)$ entspricht Identität

Anhand obiger Definition des Fehlersignals δ ergibt sich für die 1.Ableitung von $f_j(net_j)$:

$$f'_j(net_j) = f_j(net_j)(1 - f_j(net_j)) = o_j(1 - o_j)$$

$$\delta_j = \begin{cases} o_j(1 - o_j) \cdot (t_j - o_j) & j \text{ ist Ausgabezelle} \\ o_j(1 - o_j) \cdot \Sigma_k(\delta_k \cdot w_{jk}) & j \text{ ist verdeckte Zelle} \end{cases}$$

3.10 Die Systemumgebung

Die Systemumgebung umfasst die Ein-/Ausgabecodierung sowie weitere Instanzen der Daten-, Prozeß- und Schnittstellenverwaltung. Auf diese Problematik wird in späteren Kapiteln im Rahmen der Toolentwicklung (KNN 1.0) noch näher eingegangen werden.
Einige Netztypen sind nicht für die Abbildung beliebiger Vektoren geeignet, so dass gegebenenfalls Vorabtransformationen (Normalisierungen) durchgeführt werden müssen, um das jeweilige Netz in geeigneter Form trainieren zu können.

4 Software-Simulatoren

4.1 Auswahl von Software-Simulatoren

Die Forschung nach geeigneten Simulatoren unter einer „Windows-Oberfläche“ beschränkt sich auf die Suche im Internet unter Verwendung des eingesetzten Browsers „Netscape Navigator 4.0“.

Die Suche läßt sich in folgende Bereiche unterteilen:

- Suchbäume

  Innerhalb der zur Verfügung stehenden Suchbäume wie z.B. „Yahoo.de /.com“, „Kolibri“ und „Alladin“ wird zu Beginn eine globale Suche nach den Oberbegriffen gestartet. Da durch eine solche Suche eine nicht überschaubare Anzahl an Einträgen und Dokumenten aufgelistet werden, kann der Suchraum durch Verwendung von Unterbegriffen und den Grundoperatoren („UND“, “ODER“) so weit eingeschränkt werden, dass die erzielten Ergebnisse eine effiziente Beurteilung zulassen.

- Universitäten

  Die Recherche nach Forschungs- bzw. Diplomarbeiten auf dem Sektor der Simulatoren im Bereich der Medizin/Fachbereich Neurochirurgie konzentriert sich auf alle Fakultäten, die im Fachbereich Informatik, Neuro-Informatik oder Medizinische Informatik tätig sind.

- Neural Network Group

  Bei diesem Forum handelt es sich um ein im Internet publizierendes Gremium im Fachgebiet Neuro-Informatik.

  Auf den Seiten ihrer Internetadresse befinden sich die aktuellen Forschungsergebnisse mit den entsprechenden „Links“ und ein Verzeichnis der auf dem Markt zur Verfügung stehenden Simulatoren mit den zugewiesenen „ftp-Servern“.

4.2 Beurteilungskriterien

Die nicht kommerziellen Simulatoren, die innerhalb der Recherche gefunden wurden, werden anhand eines Kriterienkataloges bewertet und klassifiziert.
Die Beurteilung der einzelnen Bereiche erfolgt mit Hilfe einer Notenskala von 1 bis 6. Folgende Kriterien werden anhand der Software ausgewertet:

1. Installation:
   Da es sich bei allen Softwareprodukten um auf PC-basierende Shareware handelt, werden u.a. das benutzerdefinierte Setup, die Flexibilität und die Stabilität bewertet.
   Die Systeme werden anhand der im Handbuch befindlichen Installationsanleitung gestartet und während des Setups eingestuft.

2. Benutzungsoberfläche:
   Benutzerführung, Benutzerfreundlichkeit und Einfachheit sind einige der Unterbegriffe, mit denen diese Rubrik der Beurteilung analysiert werden.
   Nach der erfolgreichen Installation werden die Programme gestartet und mit Beispieldaten gefüllt. Inwieweit diese Aufgabe vom System geführt wird, ist eine der Analysen, durch die eine Klassifikation ermöglicht wird.

3. Softwarezuverlässigkeit:
   Für ein Produkt, das in dem Bereich neuronale Netze eingesetzt werden soll, ist die Zuverlässigkeit ein wesentliches Bewertungskriterium. Sowohl die Stabilität als auch die offene Gestaltung des Systems sind Parameter für die Zuverlässigkeit eines Programms.

4. Schnittstellen:
   Unsere Forschung konzentriert sich auf Produkte, die unter einer „Microsoft-Benutzungsoberfläche“ integrierbar sind. Durch diese Fokusierung ist die Möglichkeit des Zugriffs von externen Programmen auf den Simulator unabdingbar.
   Diese Kommunikation geschieht von den externen Programmen aus, wodurch der Simulator den Status eines Berechnungstools einnimmt.
5. Neuronale Parameter:

Durch die Vielzahl an möglichen Parametern für ein neuronales Netzwerk bedingt, werden die Softwareprodukte z.B. anhand der unterstützten Topologien, der hinterlegten Lernregeln und der Übergabeparameter der integrierten Funktionen überprüft.

6. Datenbereich:
   Da die zu analysierenden Daten die unterschiedlichsten Formate und Ausprägungen besitzen können, wird getestet, welche Formate ein Produkt unterstützt und inwieweit der Definitions-, Wertebereich und die Anzahl der zu verarbeitenden Daten beschränkt ist.

7. Resultate:
   Anhand vorbereiteter Trainings- und Testdaten werden die Simulatoren im Hinblick auf die Zuverlässigkeit der Ergebnisse und deren Darstellung überprüft. So spielen die Speicherung, die graphische Darstellung und die Analysemöglichkeiten der Ergebnisse eine wichtige Rolle.

8. Referenzen:
   Referenzen werden von Anwendern eines Simulators an den Hersteller vergeben, wodurch eine Eingruppierung des Softwareprodukts in einem Marktbereich gewährleistet werden kann.

9. Dokumentation:
   Der einzusetzende Simulator muss die unterschiedlichsten Daten und Topologien neuronaler Netze verarbeiten. Damit eine sichere Bearbeitung der Testfälle möglich ist, wird jedes Produkt auf die hinterlegte Dokumentation in Form eines Textfiles, einer Hilfedatei oder Internetseiten untersucht.
   Ein wichtiger Parameter stellt die Anwendungsfreundlichkeit, d.h. die Themen im Bereich der Beispiele, Schnittstellen und Navigation innerhalb des Programms dar.

4.3 Auswertung

In Tabelle 4.1 werden die bewerteten Simulatoren incl. der Einstufungen dargestellt. Die Zahlen in den Spaltenüberschriften entsprechen den unter 4.2 beschriebenen Bewertungskriterien bzw. dessen Nummerierung.

Die Ausprägungen jeder einzelnen Kategorie pro Simulator entspricht der „pädagogischen Notenskala“ von 1 bis 6, wobei die Note 6 als ungenügend und die Note 1 als „sehr gut“ einzustufen ist.

Simulator	Release	1	2	3	4	5	6	7	8	9	Σ
AiNet	**1.24**	1	1	4	6	5	6	3	4	2	*32*
Aspirin Migraines	**6.0**	3	4	3	5	3	4	5	4	5	*36*
Atree	**2.7**	4	1	4	5	6	5	3	6	2	*36*
BioSimPC	**2.0**	3	4	3	5	3	2	2	6	3	*31*
Brain	**1.2**	2	6	3	5	4	1	4	5	2	*32*
Cascade	**1.0**	4	3	4	3	2	1	5	4	3	*29*
HavBpNet++	**2.4**	2	1	2	2	3	1	2	3	2	***18***
NefClass	**2.04**	4	3	2	3	5	3	4	4	5	*33*
Nefcon	**1.0**	5	2	3	4	4	2	3	5	3	*31*
NeoCognitron	**1.0**	2	5	2	6	2	1	5	6	4	*33*
NeuroDS	**3.1**	3	6	1	4	4	4	6	6	3	*37*
NeuroLab	**1.2**	4	4	3	3	5	5	4	4	6	*38*
Neuron	**4.1**	3	3	4	6	3	3	3	4	4	*33*
NeuroNet	**2.0**	2	4	3	5	5	2	2	6	5	*34*
NeuroStock	**2.02**	4	5	6	4	2	5	2	4	4	*36*
Nnelmos	**1.12**	1	2	5	4	2	6	6	3	2	*31*
NN Praktikum	**1.0**	3	3	3	6	2	4	4	5	5	*35*
Perceptron	**2.0**	2	5	4	6	1	3	5	5	4	*35*
SNNS	**2.41**	2	1	3	2	1	2	2	2	1	***16***
Turbo Pascal	**6.0**	1	3	3	2	1	3	2	4	3	***22***

Tabelle 4.1 Bewertungstabelle der nicht kommerziellen Simulatoren

4.4 Zusammenfassung

Aufgrund der Analyse der einzelnen Simulatoren stellt sich heraus, dass auf dem nicht kommerziellen Bereich von Softwaretools der Stuttgarter Simmulator „SNNS 4.1“ als bestes Produkt abgeschnitten hat.
Einige der zu Verfügung stehenden Simulatoren wie z.B. „AiNet“ und „Atree“ überzeugen zwar durch ihre Benutzungsfreundlichkeit und den hinterlegten Dokumentationen, bieten jedoch im Bereich der Einstellungsmöglichkeiten der Topologien und Funktionen der neuronalen Netze kaum Variationen an.
Die Simulatoren wie z.B. „Cascade“ und „Perceptron“ haben jeden verfügbaren Parameter variabel gehalten und besitzen auch innerhalb des Datenbereichs keine Beschränkung. Da sie jedoch nur mit Hilfe einer „C++-Runtime-Umgebung“ nutzbar sind, ergeben sich Probleme im Bereich der Zuverlässigkeit und der Benutzungsfreundlichkeit.
Der Simulator „Nnelmos“ stellt ein sehr gutes Lernprogramm im Bereich der neuronalen Netze dar. Mit diesem Softwareprodukt lassen sich alle Topologien und gängigen Funktionen darstellen. Die hinterlegten Dokumentationen stehen wahlweise auf Englisch und Deutsch zur Verfügung. Jedoch ist die Übergabe von externen Daten bei diesem Programm ausgeschlossen, wodurch es lediglich in Schulungsbereichen eingesetzen werden kann.

Als Ergebnis der Untersuchung stellt sich heraus, dass die Struktur der nötigen Patternfiles (siehe 12.5) vieler Simulatoren identisch ist, wodurch sich die genauere Analyse folgender Simulatoren ergibt:

- HavBpNet++
- Turbo Pascal
- Stuttgarter Simulator SNNS

5 Der Simulator HavBpNet++

5.1 Produktbeschreibung

Das Softwareprodukt „HavBpNet++“ wird von der Firma hav.Software (hav@neosoft.com) vertrieben und simuliert die Funktionen/Strukturen eines feedforward und eines recurrent Netzes mit Hilfe der Lernregel „Backpropagation“.
Das Programm wurde mit Hilfe von „Borland C++“ objektorientiert entwickelt.
Als weitere Software stellen hav.Software das Produkt „HavFmNet++“ zur Verfügung. Das Programm ist nicht als Shareware-Version erhältlich. Bei diesem Tool werden die selbstorganisierenden Karten nach Kohonen simuliert.
„HavBpNet++“ bietet im Standard bereits die Möglichkeit, verschiedene User-Applications in das Programm mit Hilfe von vorgefertigten Schnittstellen und deren DLL-Bibliotheken zu integrieren.

Die Oberfläche des Programms wird in Abbildung 5.1 dargestellt.

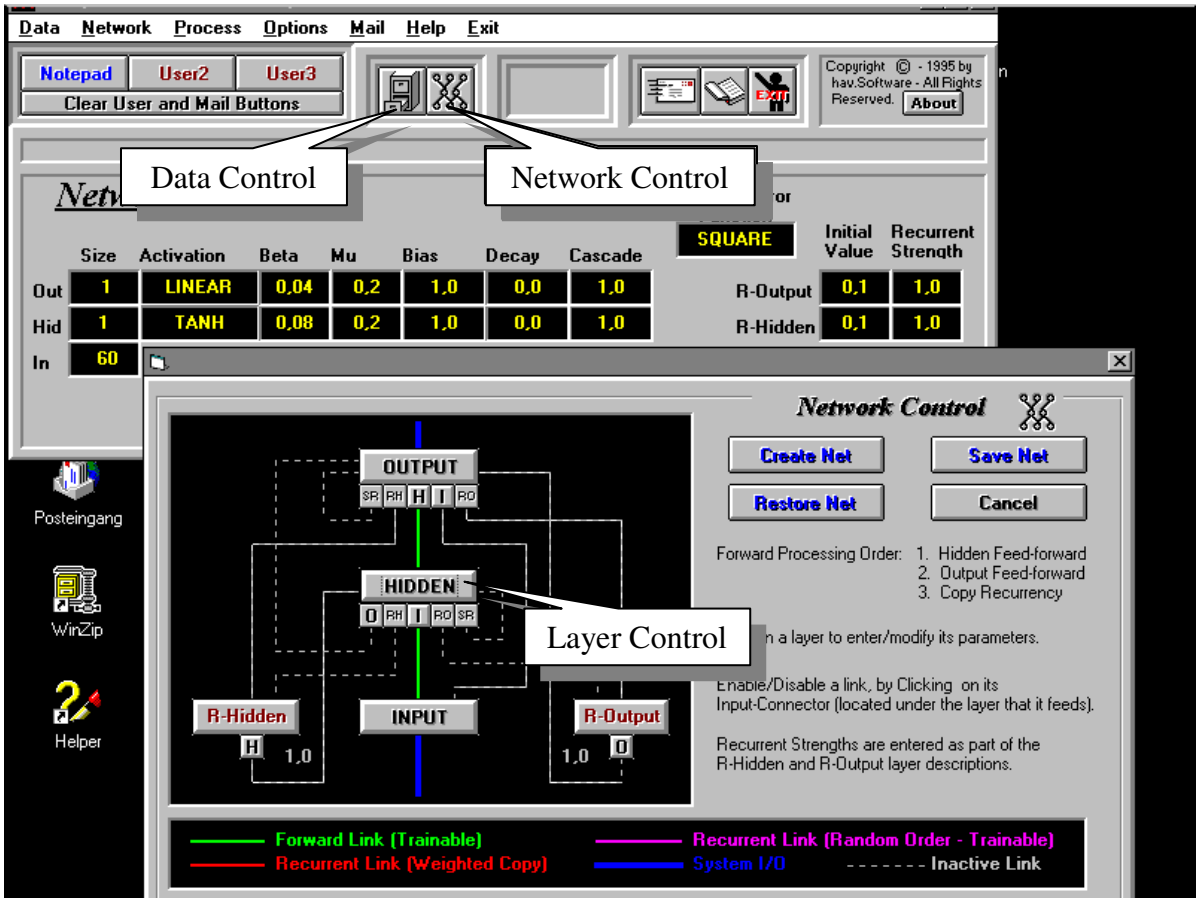


Abbildung 5.1 Benutzungsoberfläche HavBpNet++ / Network Control

5.2 Benutzerhandbuch

- Data Control

  In diesem Bereich des Programms wird die Trainings- und Testdatei selektiert. Durch Klick auf den Data-Control Button öffnet sich ein Fenster, in dem man per Doppelklick auf die entsprechenden Felder für Training und Test die Dateien auswählen kann, die dem Netz zur Verfügung gestellt werden sollen. Die Daten können dem System normalisiert oder in der Ausgangsform über eine Flag-Steuerung übergeben werden.

  Das entwickelte Tool „KNN 1.0“ bietet die Funktion, die Patternfiles automatisch zu erzeugen. Bei der Speicherung der Testdaten muss die Datei die Endung „DAT“ besitzen, damit die Datei von „HavBpNet++“ erkannt und geladen werden kann.

- Network Control

  Mit dieser Funktion kann den einzelnen Schichten des neuronalen Netzes die Anzahl an Neuronen zugeordnet werden. Die Struktur der „hidden Schicht“ kann zusätzlich anhand der Aktivierungsfunktion über eine Listbox (Bsp.: TanH, Sigmoid) verändert werden. Die benötigten Parameter, wie z.B. für die Irrtumswahrscheinlichkeit, werden mit Hilfe der Variablen „Beta“ eingestellt.

  Nach Festlegung der Struktur und der Eigenschaften des Netzes wird dieses mit Hilfe des „Create Net“-Buttons fertiggestellt. Die Speicherfunktion steht in der nicht registrierten Version nicht zur Verfügung.

- Training Control

  Nach Bestimmung der möglichen Parameter kann das Trainieren des Netzes mit Hilfe des Training Controls eingestellt werden. Diese Maske wird in Abbildung 5.2 dargestellt. Durch „Error Measurement Control“ kann die Art der Fehlerklassifizierung gewählt werden. Im Beispiel wird das Vorzeichen zur Analyse der Resultate herangezogen.

  Wird das Training gestartet, werden die Resultate graphisch und als Prozentwerte angezeigt. Das beste Ergebnis wird incl. des prozentualen Fehlers festgehalten.

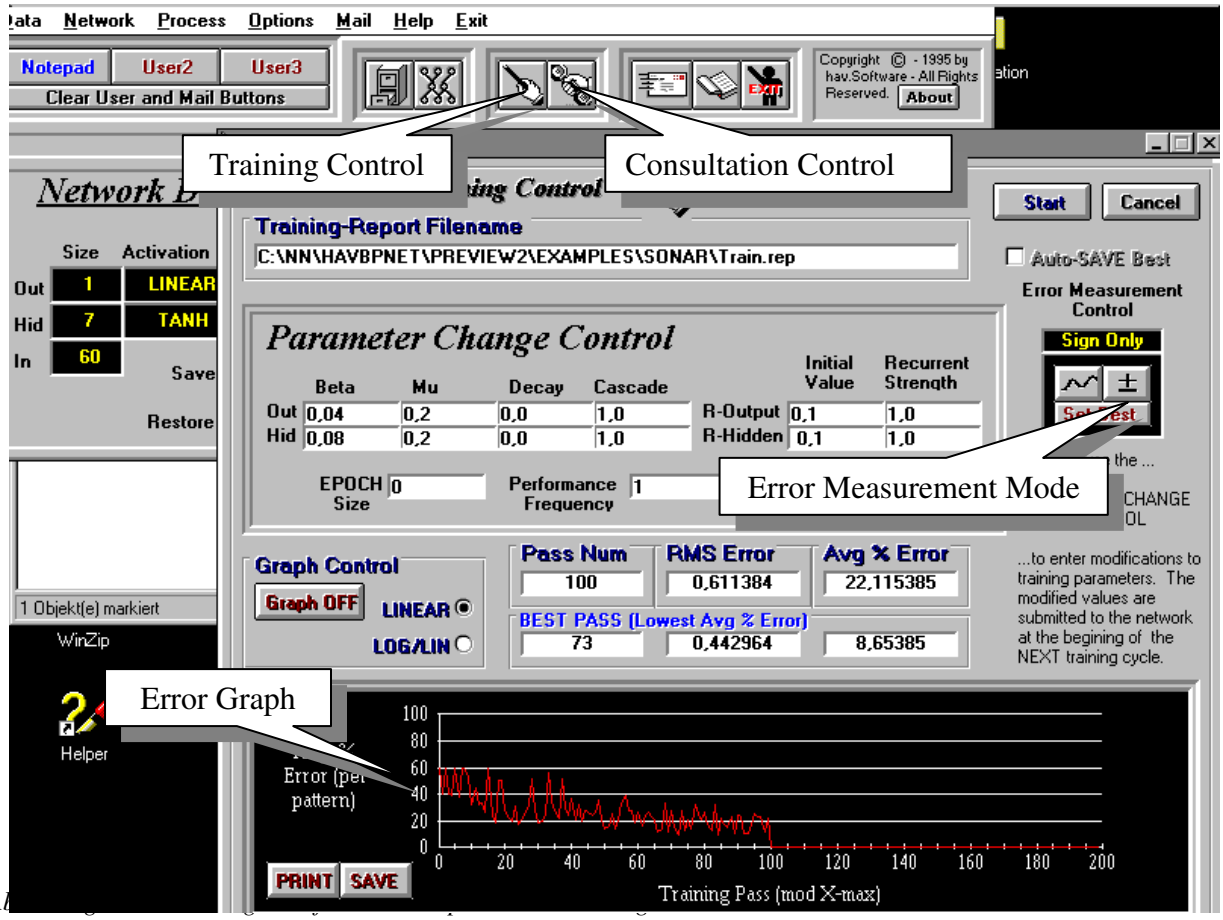
  Das in Abbildung 5.2 dargestellte Beispiel besitzt eine Zuverlässigkeit von ca. 91,4 %.

- Consultation Control

Nach erfolgreichem Trainieren des Netzes kann die errechnete Struktur getestet werden. Dies geschieht über den „Consultation Control“-Button. In dem sich öffnenden PopUp werden der Testfile und die Protokolldatei angegeben. Nach Bestätigung durch „Consult“ werden diese Testdaten dem Netz zugeführt und die Ausgabewerte berechnet.
Das Ergebnis incl. des prozentualen Fehlers wird in der Protokolldatei abgelegt. Die Datei besitzt die Endung „REP“ und hat folgende Struktur:

Pattern-Nr *Output-Ist* *Output-Soll* *Absoluter Fehler [%]*

In der letzten Zeile der Datei werden der Mittelwert, der absolute Fehler in % und der des quadratischen Fehlers gespeichert.



Abb

6 Turbo Pascal

6.1 Grundstrukturen

Die „Interdisziplinäre Arbeitsgruppe neuronale Netze“ von der Universität Mannheim entwickelte verschiedene Turbo Pascal Files, die es ermöglichen, einige Topologien und Strukturen künstlicher neuronaler Netze abzubilden.
Die Dateien sind ab Turbo Pascal Version 5.5 einsetzbar. Vor dem ersten Start der Simulatoren müssen die hinterlegten Units erzeugt werden. Die entsprechenden Dateien werden durch „NN_*.PAS“ klassifiziert und können, nach Einstellung der „Compiler Destination“, auf Disk in Turbo Pascal Unit Files (Endung „TPU“) konvertiert werden.
Die Verzeichnisse des Programms müssen innerhalb der Bereiche „EXE & TPU Directories“ und „UNIT Directories“ durch die erzeugten Unit-Verzeichnisse ergänzt werden.
Einige dieser Simulatoren-Dateien verlangen die explizite Eingabe der Dateinamen der Patternfiles, des BGI-Pfades und der Gewichtedatei. Diese Angaben werden direkt in den Quellcode an die entsprechenden Stellen geschrieben.
Die für die Simulation nötigen Konstanten, wie z.B. die Anzahl der Neuronen oder die Anzahl der Zyklen, werden in jeder Pascal-Datei im Deklarationsteil „const“ vereinbart.
Beispiel der Konstantendeklaration:

```
i_number_of_cycles = 32000;      Anzahl der Lernzyklen
i_dim_inputvector_xs = 16;       Anzahl der Komponenten von xs
i_dim_inputvector_xa = 13        Anzahl der Komponenten von xa
i_number_of_neurons = 100        Gesamtanzahl der Neuronen
i_number_of_x_neurons = 10       Anzahl der Neuronen in x-Richtung
i_number_of_y_neurons = 10       Anzahl der Neuronen in y-Richtung
i_max_learnrate = 0.9            Anfangsgröße der Lernrate
i_min_learnrate = 0.1            Untergrenze der Lernrate
i_map_update = 1                 Bildschirmanzeige nach x Lernzyklen
s_patternfile = 'neuron.pat'     Dateiname der Eingabemustermatrix
s_netfile = 'neuron.net'         Dateiname der Verbindungsstruktur
s_bgi_path = 'c:\tp\bgi'         Pfadname des BGI-Treibers
```

6.2 Die Simulatoren

Für die Untersuchung der Daten des Städtischen Klinikums Fulda werden die sich anschließenden Pascal-Files verwendet und näher beschrieben:

- BLZ_CREA.PAS:

  Nach Starten dieses Programms wird eine zufällig bestimmte Gewichtedatei erzeugt. Nach Angabe des zu vergebenden Dateinamens werden zudem die Anzahl der

Knoten aus einem Wertebereich von [2;30], der Vernetzungsgrad im Prozentformat (Menge der natürlichen Zahlen) und der maximale Vernetzungsgrad aus der Menge der ganzen Zahlen vom Benutzer eingetragen.
Die Struktur der konvertierten Gewichtungsdatei kann anhand des Pseudocodes erkannt werden:

```
„Anzahl der Elemente“ := „Anzahl der Knoten“
For i = 1 to „Anzahl der Knoten“
    For j = 1 to „Anzahl der Elemente“
            Gewicht[j] := Random [0;“Maximale Verbindungsgewichte]
    J := j + 1              {Next j}
    „Anzahl der Elemente“ := „Anzahl der Elemente“ –1
    i := i + 1              {Next i}
```

Durch die beschriebene Struktur wird als Beispiel folgende Datei erzeugt.

```
Anzahl der Knoten          10
Vernetzungsgrad            7
Max. Verbindungsgewichte   9

0   5   0   4   6   6   2   8   0   7
0   0   9   3   0   0   2   5   4
0   2   7   1   0   7   0   2
0   7   6   6   5   0   3
0   1   0   9   7   4
0   0   3   8   0
0   4   2   0
0   0   0
0   1
0
```

- BLZ_COMP.PAS

  Sofern eine Gewichtungsdatei existiert, kann mit diesem Tool der maximale Schnitt eines Graphen durch vollständige Enumeration bestimmt werden. Voraussetzung

zum Starten sind die Eingabe der Anzahl der Knoten und der Dateiname der Gewichtungsdatei.

Für das obige Beispiel ergibt sich folgender maximaler Schnitt:

Knoten in V0: *1 3 4 9*
Knoten in V1: *2 5 6 7 8 10*
Max. Cut-Summe: *110*

- BLZ_MAXC.PAS:

  Dieses Tool berechnet ebenso wie das Programm „BLZ_COMP.PAS“ den maximalen Schnitt eines Graphen anhand der gleichen Eingabeparameter.
  In diesem Fall wird das Ergebnis mit Hilfe der Boltzmann-Maschine errechnet.

- BPG_ASCI.PAS:

  Über diese Datei ist es möglich eine Mustererkennung anhand eines erzeugten Patternfiles (muss im Quellcode angegeben werden) zu simulieren.
  Nach Start des Programms steht ein Menü mit den aufgeführten Punkten zur Verfügung.

  1 Laden eines gelernten Netzes
  2 Lernparameter verändern
  3 Trainieren
  4 Testen
  5 Sichern des aktuellen Netzes
  6 Ende

- BPG_XOR.PAS:

  Anhand dieses Programms kann das Trainieren eines XOR-Musters simuliert werden. Bei dem Lernalgorithmus handelt es sich um „Backpropagation“. Voraussetzung ist der zu trainierende Patternfile, der dem System ohne Pfadangabe im Quellcode zu übergeben ist.

- INT_UNDE.PAS:

  Das Textverstehen eines interaktiven Netzes kann unter der Verwendung des Textfiles (*.NAM) und einer Gewichtungsdatei (*.WGT) generiert werden. Die

Endungen der Dateien können benutzerdefiniert bestimmt werden, sofern das Textformat eingehalten wird.

Die Erkennung des Musters/Textes wird von dem Programm graphisch dargestellt.

- GDS_NQUE.PAS:

  Durch die Struktur eines „Guarded-Discrete-Stochastic-Netzwerks“ wird das „N-Damen-Problem“ gelöst. Damit die Lösung auch graphisch dargestellt werden kann, muss im Quellcode der hinterlegte BGI-Pfad angepasst werden.

  Die graphische Interpretation des Ergebnisses zeigt Abbildung 6.1 bei folgenden Parametern:

  Anzahl der Damen: *12*

  Abbruchkriterium: *5*

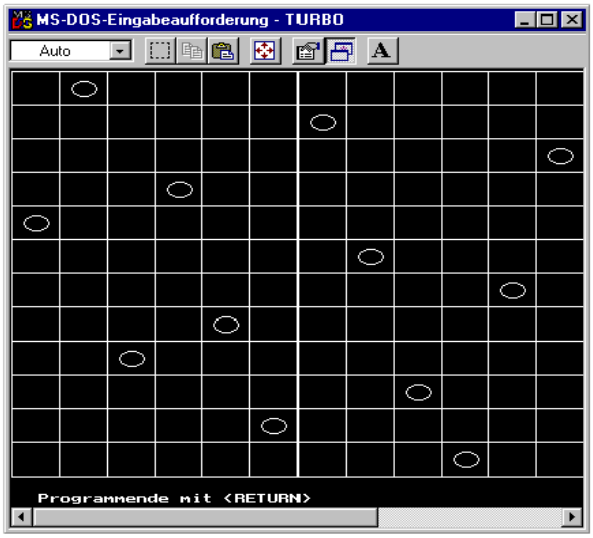


Abbildung 6.1 Graphik des N-Damen-Problems

- HOP_ASSO.PAS:

Mit Hilfe dieser Datei lässt sich das Verhalten eines Hopfield-Netzes als Musterassoziator simulieren. Es dient der Rekonstruktion fragmentierter und verzerrter Eingabemuster.

Um das Ergebnis auch graphisch darstellen zu können, müssen die Angaben des BGI-Pfades und der Testdatei im Deklarationsteil des Programms ergänzt werden.

- KOH_TIER.PAS:

  Dieser Simulator stellt die Funktionen der selbstorganisierenden Karten nach Kohonen zur Verfügung.

  In dem eigentlichen Programmcode müssen die Angaben bzgl. des BGI-Pfades, der Testdatei (*.DAT) und der abzuspeichernden Netzdatei (*.NET) eingetragen werden.

  Nach Start des Programms wird die Netzdatei entsprechend der eingestellten Zyklen erzeugt und abgespeichert.

7 SNNS 4.1

7.1 Systemvoraussetzung

„SNNS 4.1“ ist ein Simulator für künstliche neuronale Netze, welcher am „Institut für Parallele und Verteilte Höchstleistungsrechner, IPVR“ der Universität Stuttgart seit 1989 entwickelt wird. Die unterstützten Systeme können sowohl auf UNIX als auch auf Microsoft Windows 95/NT basieren.

Die in dieser Diplomarbeit verwendete Version des Stuttgarter neuronale Netze Simulators (SNNS 4.1) ist das Release V4.1 für Microsoft Windows 95/NT. Diese wurde gewählt, damit die Plattformkonformität zwischen den Städtischen Kliniken Fulda und der Fachhochschule Fulda gewährleistet ist.

Um die Software in Betrieb nehmen zu können, ist es erforderlich, auf dem Windowssystem zuvor ein X-Win32 zu installieren und zu starten. In einem Client-Server-Netzwerk wird dem Stuttgarter Simulator dadurch eine feste Adresse (IP-Adresse) zugewiesen.

Wird der Simulator auf einem Stand-Alone-System betrieben, so muss gewährleistet werden, dass durch den Start des X-Win32-Programms eine DFÜ-Netzwerkverbindung (z.B. via Internet) aufgebaut wird, um somit die Zuweisung der Adresse erreichen zu können.

7.2 Benutzungsoberfläche

Nach dem erfolgreichen Start des Simulators erscheint dem Anwender das „SNNS 4.1 Manager Panel“. Es handelt sich hierbei um die Grunddialogbox des „SNNS 4.1“, von der aus man per Buttonclick in alle wichtigen funktionalen Untermenüs gelangen kann.

Im folgenden werden die wichtigsten Funktionstasten mit der Beschreibung der Ereignisse aufgezeigt.

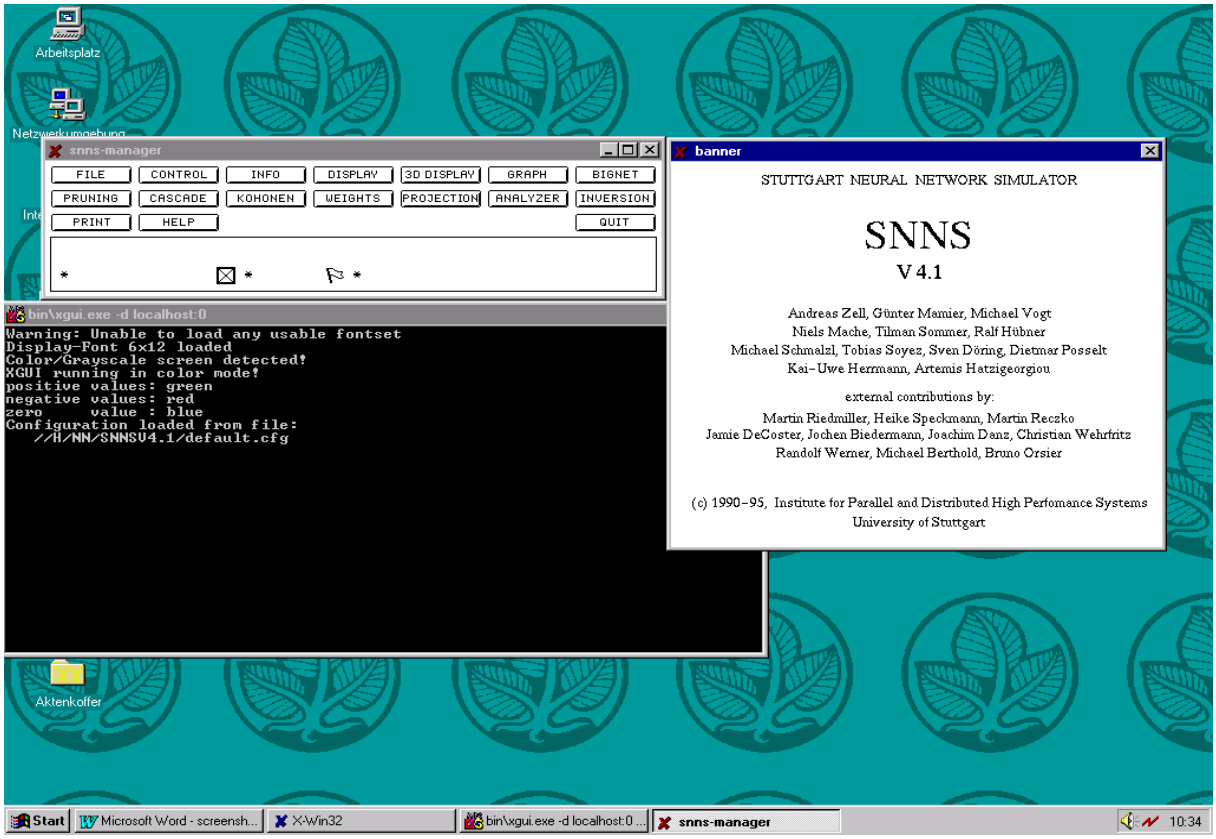


Abbildung 7.1 SNNS Benutzungsoberfläche

HELP - Button, „Manager Panel“:

Die Hilfefunktion ist mit internen Links versehen, die ein einfaches Benutzen der Hilfe ermöglichen.

BIGNET - Button, „Manager Panel“:

Funktionsschalter, um ein neues künstliches neuronales Netz kreieren zu können. „SNNS 4.1“ermöglicht dem Benutzer, viele unterschiedliche Netzwerke verschiedener Topologien zu erzeugen. Als Beispiel sei hier ein „feedforward“ aufgezeigt, welches auch zur Auswertung der gelieferten Klinikdaten herangezogen wird.

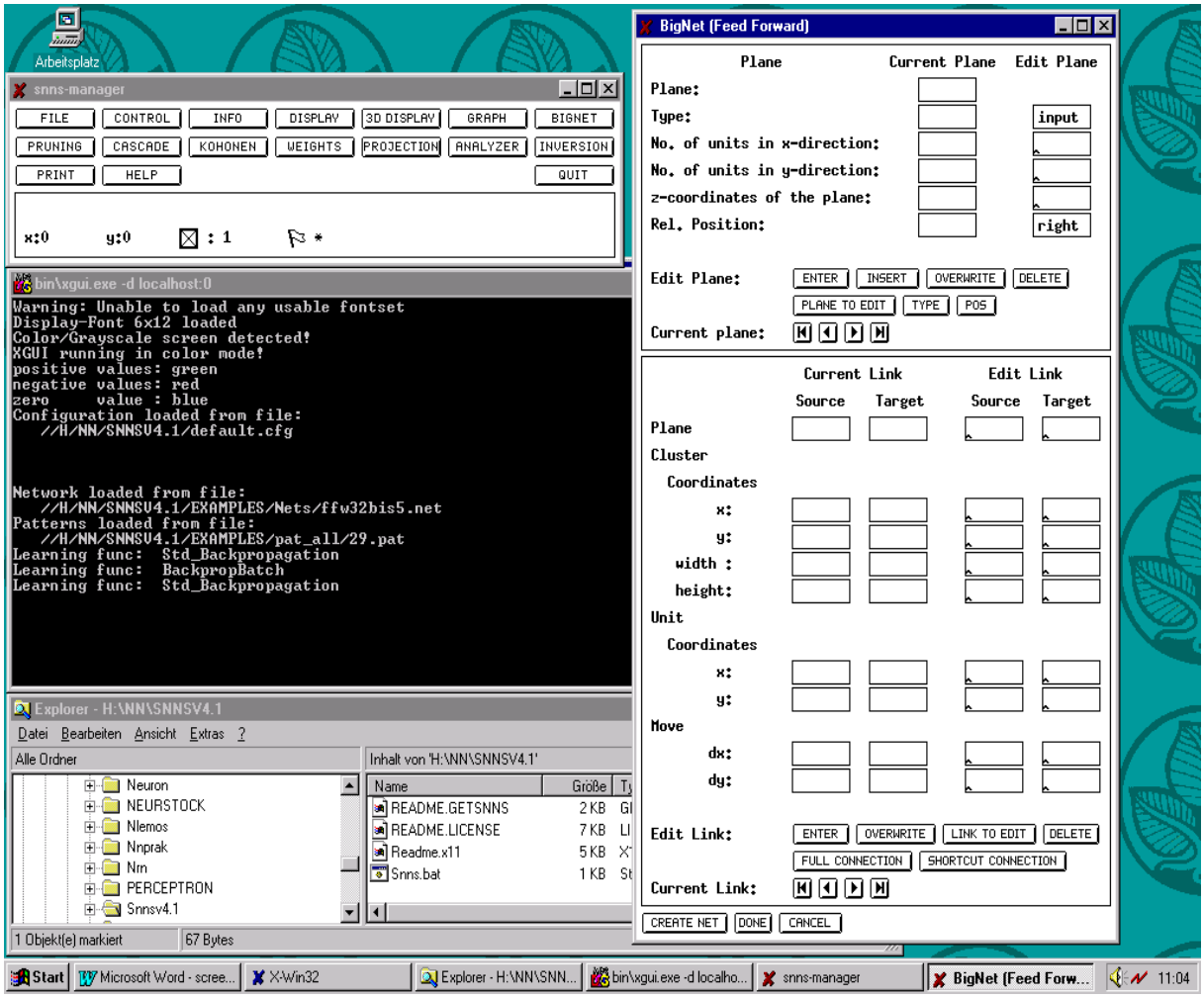


Abbildung 7.2 SNNS Netzgenerator

Zuerst muss eine Eingabeschicht definiert werden; dazu muss das Feld „TYPE“ sooft angeklickt werden, bis im oberen rechten Fenster „input“ erscheint. Die darunter liegenden Eingabefenster dienen der Angabe zur Layer-Topologie. Hier muss die Anzahl der Einheiten in ein karthesisches 3-dimensionales Koordinatensystem eingetragen werden. Will man eine nur 2-dimensionale Schicht erzeugen, so ist mindestens in einer Richtung eine Eins einzutragen.

Der Arbeitsablauf erfolgt identisch für verborgene (hidden) Schichten– „TYPE“ muss angeklickt werden, damit oben rechts „hidden“ erscheint – und analog für die Ausgabeschicht – oben rechts muss „output“ aktiviert sein.

Über den „POS“ - Button kann man die relative Lage der Schicht (außer der ersten) im Netz angeben.

Für das Netzwerk können die Verbindungen zwischen den Neuronen entweder individuell oder vorbereitet, zum Beispiel voll verschaltet („FULL CONNECTION“), gewählt werden.
Danach muss das Netz durch Klicken auf den „CREATE“ - Button erstellt werden. Ist noch ein vorheriges künstliches neuronales Netz aktiv, so wird das Alte bei Bestätigung durch die neue Struktur überschrieben.
Für die anderen Netztopologien erscheint ebenfalls bei der entsprechenden Auswahl ein selbsterklärendes Fenster.
Das Speichern oder Laden von neuronalen Netzen erfolgt über den „FILE“ – Button.

FILE - Button, „Manager Panel“:

Unter diesem Button verbirgt sich der Dateimanager des „SNNS 4.1“. Es werden fünf Dateiformate unterstützt, die geladen oder gespeichert werden können. Durch Mausklick auf das gewünschte Aktionsfeld können die einzelnen File-Operationen getätigt werden.

Aktionsfeld „NET“: Dieses Feld bezieht sich auf die Netzwerkdefinitionsdateien, die Informationen über Netzwerktopologie und Lernregeln beinhalten. Die Dateien haben die Form „*.NET“.

Aktionsfeld „PAT“: Dieses Feld dient dazu, die Patternfiles laden zu können, die von der Gestalt „*.PAT“ sind. Patternfiles können sowohl Trainingsdaten als auch Testdaten beinhalten.

Aktionsfeld „RES“: Die Abkürzung RES steht für Resultfiles (Ergebnisdateien), die mit der Endung „*.RES“ abgespeichert werden. Es ist möglich, sowohl das Startpattern als auch das Endpattern anzugeben. Es kann weiterhin der Modus „CREATE“ (Erzeugen) oder der Modus „APPEND“ (Anhängen) ausgewählt werden. Das hat zur Folge, dass im ersten Fall eine neue Datei erzeugt wird und die alte Datei, falls der Dateiname schon vergeben war, überschrieben wird. Im zweiten Fall wird das neue Ergebnis an die vorhandene Datei angehängt. Input und Output können getrennt per Umschalter eingeblendet oder ausgeblendet abgespeichert werden.

Nach einer Speicheraktion ist besonders darauf hinzuweisen, dass auf Rechnersystemen, die mit einem C++-Compiler ausgestattet sind, die Dateierweiterungen der Ergebnisfiles vor dem Öffnen zum Lesen vorher umbenannt werden müssen. Als Beispiel sei hier genannt: „*.RES“ für Resultate umbenennen in „*.ERG“ für Ergebnis.
Dies ist notwendig, da es sonst zu Systemabstürzen kommen kann, weil die Datei unter Umständen von dem C++-Compiler geöffnet wird. Die Endung „RES“ wird durch das Programm „Borland C++“ reserviert.

Aktionsfeld „CFG“: Hier können Setup-Parameter und Schichtenbezeichnungen geladen und gespeichert werden. Es ist möglich, eine Konfiguration für mehrere verschiedene Netzwerke zu definieren. Als Standard wird die Datei „default.cfg“ automatisch geladen.

Aktionsfeld „TXT“: Hier kann eine Protokolldatei erzeugt werden, die Dateioperationen, Wertdefinitionen und Lernprotokoll beinhaltet.

CONTROL- Button, „Manager Panel“:
Jedes Training und Testen der Daten erfolgt im „SNNS 4.1“ über das Control–Panel, wobei sich dieses in zwei Teilen darstellt.
Die obere Hälfte steuert die den Trainingsprozeß definierenden Parameter, wie Schrittweite, Anzahl Lernzyklen und Patternnummer.
Die unteren vier Zeilen beziehen sich auf Lern-, Update-, Initialisierungs- und Remap-Funktionen.

Initialisierung des Netzes:
Um ein Netz zu initialisieren, wird der „INIT“ –Button benötigt, der sich in der ersten Zeile des Control – Panels befindet.

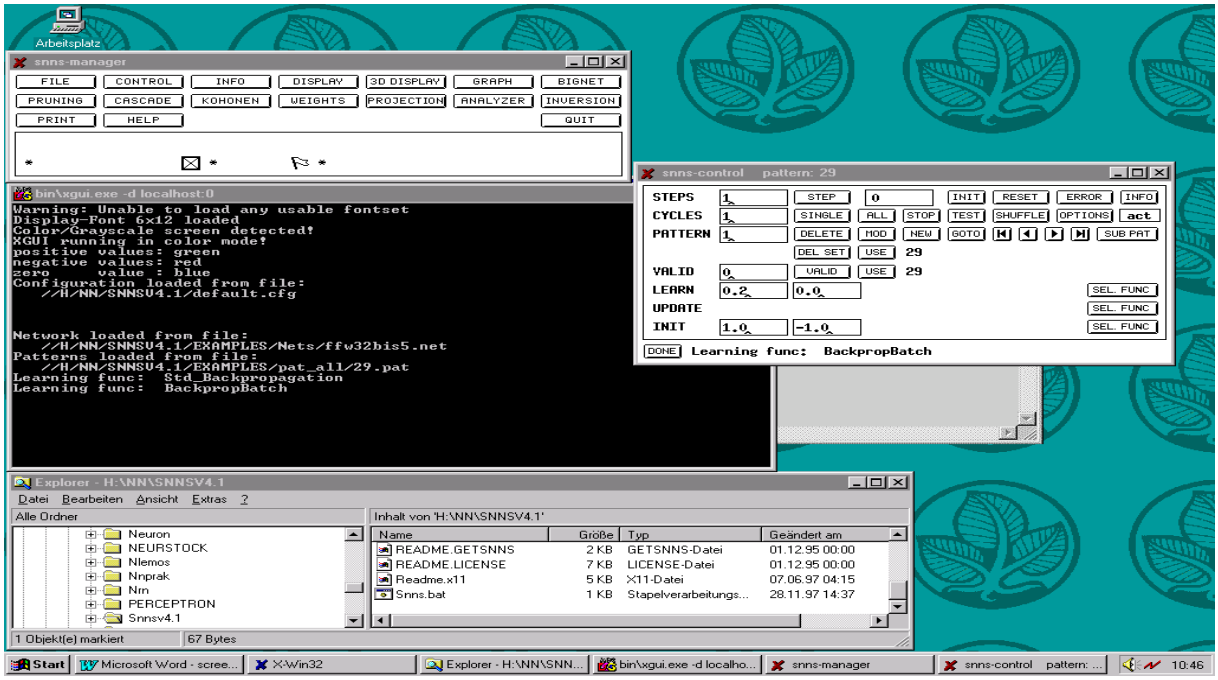


Abbildung 7.3 SNNS control panel

Wählen einer Lernfunktion:

Die Grundeinstellung für „feedforward“ -Netze ist die „Std_Backpropagation“ -Regel. Durch Anklicken des Buttons für Lernfunktionen erscheint ein fly-out Menü mit verschiedenen Lernfunktionen, die gewählt werden können.

Nachdem verschiedene andere Lernfunktionen (Momentum etc.) ebenfalls getestet wurden, aber keine besseren Ergebnisse geliefert haben, wird die Standardeinstellung für die ausführlicheren Tests übernommen.

Trainieren des Netzes:

Sind alle Parameter gesetzt, so kann damit begonnen werden das Netz zu trainieren. Um das Training zu starten, wird der „ALL“ –Button aktiviert, das Netz durchläuft in der Lernphase die in “CYCLES“ angegebene Anzahl von Lernzyklen. Die Lernphase kann über den „STOP“ – Button abgebrochen werden.

Die mittleren quadratischen Fehler werden in einem Fenster kontinuierlich angezeigt.

Im Controlpanel befinden sich auch zwei Buttons mit der Bezeichnung „USE“, welche es dem Anwender erlauben, bestimmte Datensätze auszuwählen, die für das Training und für die Validation relevant sind.

Graphische Darstellungsmöglichkeiten im Manager Panel

Die graphische Darstellung der Netze erfolgt im „SNNS 4.1“ über die zwei Schaltflächen „DISPLAY“ und „3D DISPLAY“ im Managerpanel. Verbindungen zwischen den Neuronen können global entweder ein- oder ausgeblendet werden.

Die zwei folgenden Abbildungen zeigen jeweils einen Ausschnitt des „feedforward“-Netzes mit dem Aufbau:

3 *Eingabeneuronen (links oben)*

32 *Neuronen in der 1. verborgenen Schicht*

16 *Neuronen in der 2. verborgenen Schicht*

8 *Neuronen in der 3. verborgenen Schicht*

5 *Neuronen in der 4. verborgenen Schicht*

3 *Ausgabeneuronen*

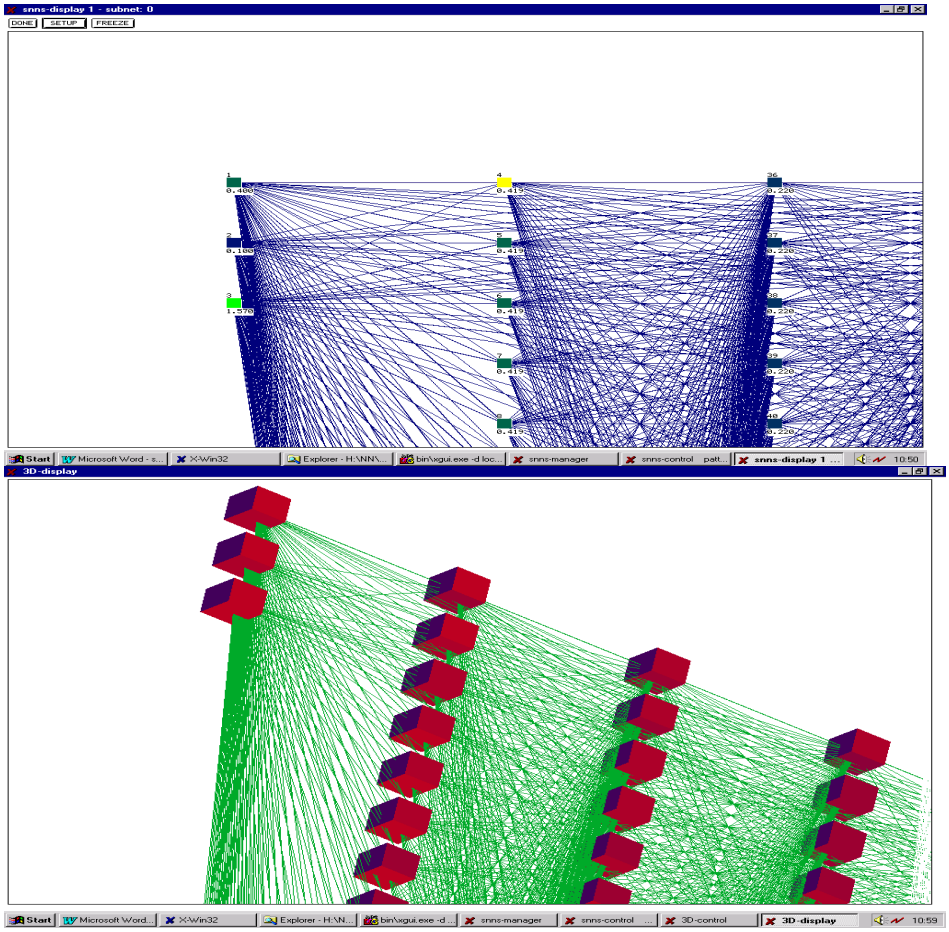


Abbildung 7.4 SNNS graphische Netzwerkdarstellung

Ein detailliertes Handbuch zum Stuttgarter neuronale Netze Simulator (SNNS 4.1) kann via Internet vom Server der Universität Stuttgart als Postscriptfile heruntergeladen werden, es hat einen Umfang von 345 Seiten im DIN A4 Format.

8 Multimodales Neuromonitoring in der Neurochirurgie

8.1 Projektbeschreibung

Das Projekt „Multimodales Neuromotoring“ des Klinikums für Neurochirurgie des Städtischen Klinikums Fulda lässt sich in die Bereiche der Datenerfassung und der Datenverarbeitung untergliedern.

Die in diesem Projekt intergrierten Teilbereiche und deren Beziehungen untereinander werden in Abbildung 8.1 graphisch dargestellt.

Für diese Arbeit relevante Unterprojekte sind zum einen die Analyse/Berechnung und zum anderen die Entscheidungsfindung durch neuronale Netze. Sie werden in der Abbildung 8.1 im Sektor Ziele beschrieben.

Innerhalb des Teilprojekts „Hardwarelösungen“ wird das Monitoring realisiert und die erzeugten Daten, ergänzt durch beispielsweise das Expertenwissen und weitere klinische Fakten, dem MNM-PC (multimodaler Neuromotoring PC) zur Verfügung gestellt.

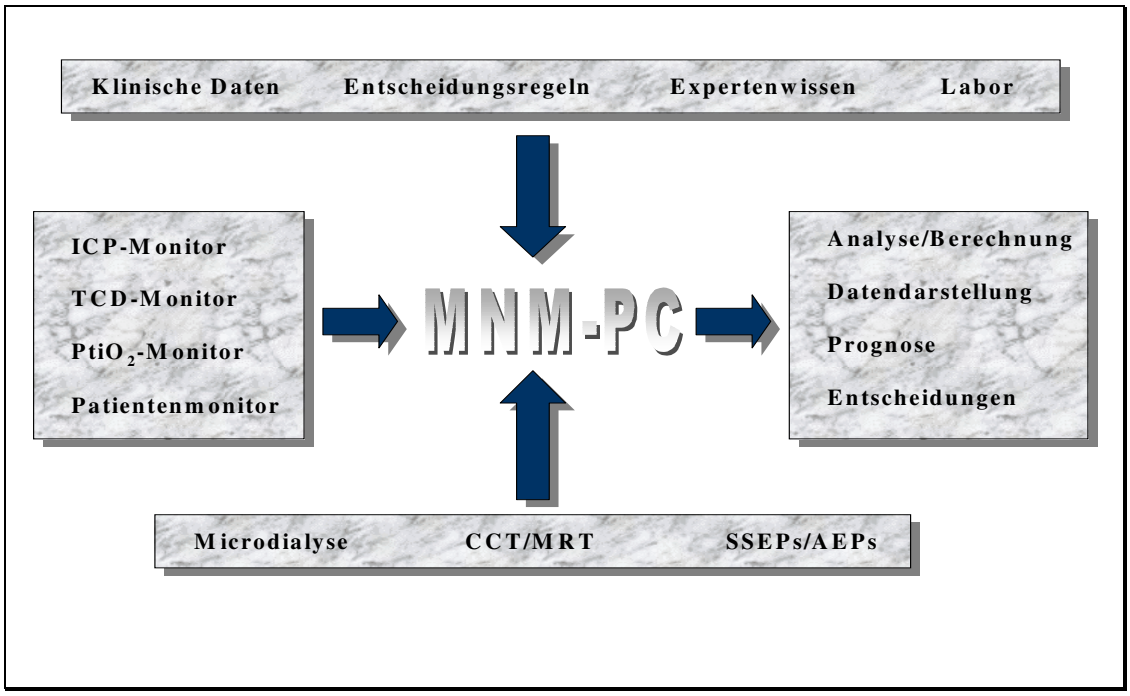


Abbildung 8.1 Übersicht aller Systeme zum multimodalen Monitoring

8.2 Datenerfassung

Innerhalb des Teilprojekts der Datenerfassung ist es wichtig, dass alle separat erfassten Daten zusammengeführt werden. Es soll ein MNM-PC entwickelt werden, der die unterschiedlichen Daten, die durch Tabelle 8.1 beschrieben werden, in ihren Ausprägungen und verschiedenen Formaten über die zur Verfügung stehenden Messapparaturen in einem System abbildet.
Die Messungen basieren zum Teil auf der Tatsache eines erhöhten intrakraniellen Druckes. Die Messwerte werden beispielsweise mit Hilfe einer Hirnsonde (parachymatös oder ventrikular) oder über die Magnetresonanztomographie (MRT) bzw. Computertomographie (CCT) ermittelt.

Die Messbereiche lassen sich wie folgt unterscheiden:

- *Neurologische Daten*:
  Pupillen, Lichtreaktion (dir, kons), Hirnstammreflexe, Dezerebrationszeichen

- *Messtechnische Daten*:
  ICP, CPP, ptiO_2, PtiCO_2, pH, Temperatur transkranieller Doppler

- *Laborchemische Daten*:
  Mikrodialyse, arterio-venöse O_2/CO_2-Differenzen, Lactat im Liquor

- *Anästhesiologische Daten*:
  Vitalparameter

- *Kardiologische Daten*:
  Herzfrequenz, -Rhythmus, invasive Drücke (arteriell, venös)

- *Respiratordaten*:
  Beatmungsparameter, BGA-Daten (pO_2, pCO_2, pH, BE usw.)

Pos	Eingang	Einheit	kontinuierlich	Schnittstelle

1.	ICP	mmHg	Ja	analog
2.	pO_2	mmHg	Nein (Paratrend 7: Ja)	manuell digital/analog
3.	pCO_2	mmHg	Nein (Paratrend 7: Ja)	manuell digital/analog
4.	pH	-	Nein (Paratrend 7: Ja)	manuell digital/analog
5.	$PtiO_2$	mmHg	Ja	digital/analog
6.	$PtiCO_2$	mmHg	Ja	digital/analog
7.	tipH	-	Ja	digital/analog
8.	tiTemp	°C	Ja	digital/analog
9.	Herzfrequenz	/min	Ja	digital/analog
10.	art Druck (syst/diast)	mmHg	Ja	digital/analog
11.	ven. Druck	mmHg	Ja	digital/analog
12.	TCD: Syste	cm/sec	Ja	digital
13.	TCD: Diast	cm/sec	Ja	digital
14.	TCD: Mean	cm/sec	Ja	digital
15.	TCD: PI	-	Ja	digital
16.	TCD: RI	-	Ja	digital
17.	TCD: Linnegrad	-	Nein	manuell
18.	TCD: CO_2-Reaktiv.	-	Nein	manuell
19.	Resp.: Freq	/min	Nein (Ja)	manuell digital/analog
20.	Resp.: FiO_2	%	Nein (Ja)	manuell digital/analog
21.	Resp.: Beatm Muster	-	Nein	manuell
22.	Resp.: PEEP	mmHg	Nein (Ja)	manuell digital/analog
23.	CCT	-	Nein	manuell
24.	MRT	-	Nein	manuell
25.	Median. SEP (li/re)	-	Nein	manuell
26.	Tibial. SEP(li/re)	-	Nein	manuell
27.	AEP (re/li)	-	Nein	manuell
28.	NIRS	%	Nein (Ja)	manuell digital/analog
29.	Microdialyse-Lactat	mg/dl	Nein	manuell
30.	Labor (Hb, Hkt...)	mg/dl,%	Nein	manuell

Tabelle 8.1 Datenbeschreibung der Messwerte des MNM

Die Zusammenhänge der einzelnen Messwerte lassen sich aus Abbildung 8.2 entnehmen. Die dargestellten Beziehungen entstehen durch das multimodale Neuromonitoring und sind durch ein Regelwerk klar definierbar.

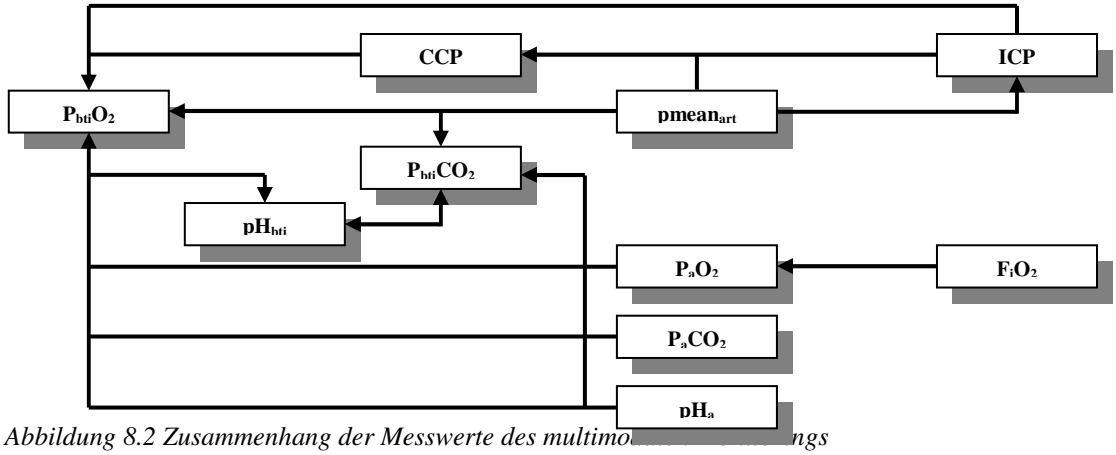


Abbildung 8.2 Zusammenhang der Messwerte des multimo[illegible]ngs

8.3 Datenverarbeitung

Die über das multimodale Neuromonitoring erzeugten Daten werden mit Hilfe der statistischen Algorithmen und nicht kommerzieller Simulatoren neuronaler Netze verarbeitet und analysiert.

Aufgrund der in Tabelle 8.1 beschriebenen Messwerte lassen sich beliebige Kombinationen der Werte erzeugen. Die dadurch bedingten Zusammenhänge werden mit Hilfe der Regressions-/Korrelationsanalyse voruntersucht und anschließend für die Prognose/Diagnostik über ein neuronales Netz abgebildet.

Voraussetzungen für eine solche zukunftsorientierte Aussage sind ein fundiertes Expertenwissen, hinterlegte Entscheidungsregeln und Messwerte, die innerhalb eines plausiblen Wertebereiches gespeichert werden.

Eine mögliche Prognose der Glasgow Outcome Scores aufgrund der Messwerte der evozierten Potentiale , des Glasgow Coma Scores (GCS) und graphischen Interpretationen aus den Bereichen der Computertomographie (CCT) und der Magnetresonanztomographie (MRT) wird in Abbildung 8.3 dargestellt.

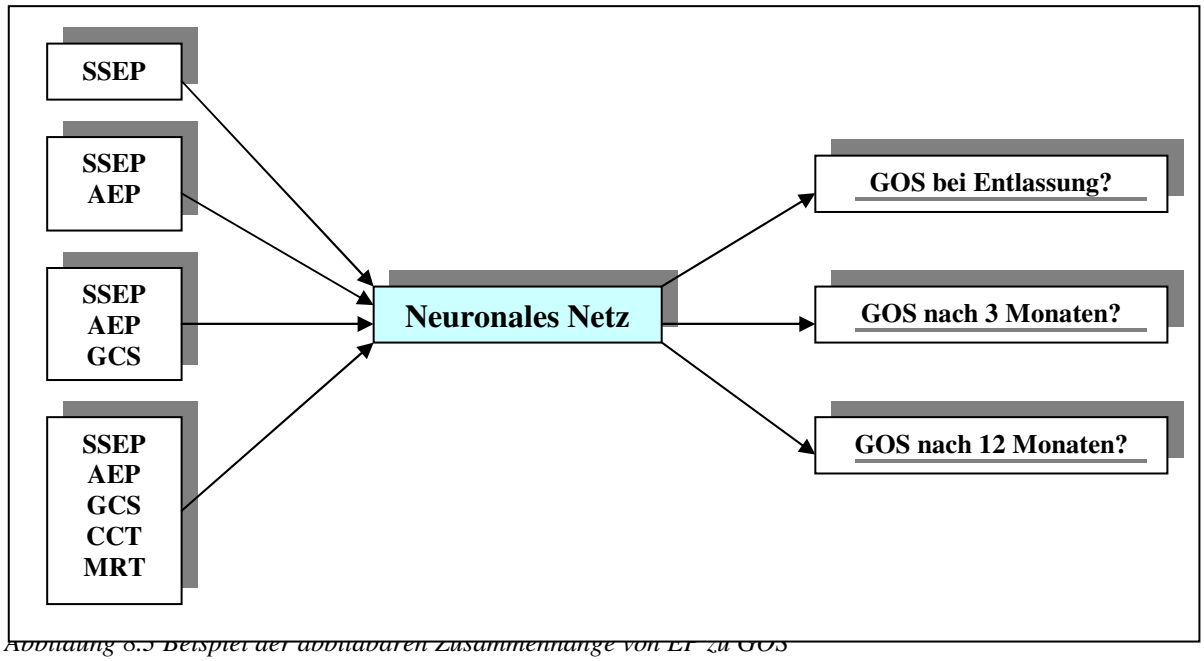


Abbildung 8.3 Beispiel der abbildbaren Zusammenhänge von EP zu GOS

Die Interpretation der gelieferten Daten und Resultate ist stets unter Berücksichtigung des Datenvolumens und der Datenreduzierung aufgrund des Wertebereiches zu sehen.

Die Werte des evozierten Potentials und der Glasgow Outcome Scores werden mit Hilfe einer natürlichen Zahlenskala ($0 < x < 6$) normiert.

Damit eine Prognose nicht nur aufgrund der zur Verfügung stehenden Daten getroffen wird, muss das Expertenwissen in Form einer Wissensbasis hinterlegt sein.

Durch diese Fakten und Regeln ist es möglich einen Ablaufgraphen der Entscheidungsfindung zu erstellen, der in das künstliche neuronale Netz zu integrieren ist.

In Abbildung 8.4 wird exemplarisch ein solcher Entscheidungsbaum zur Hirndrucktherapie dargestellt. Die einzelnen Einflussfaktoren können an jeder Stelle des Baumes die Prognose/Therapie verändern bzw. beeinflussen.

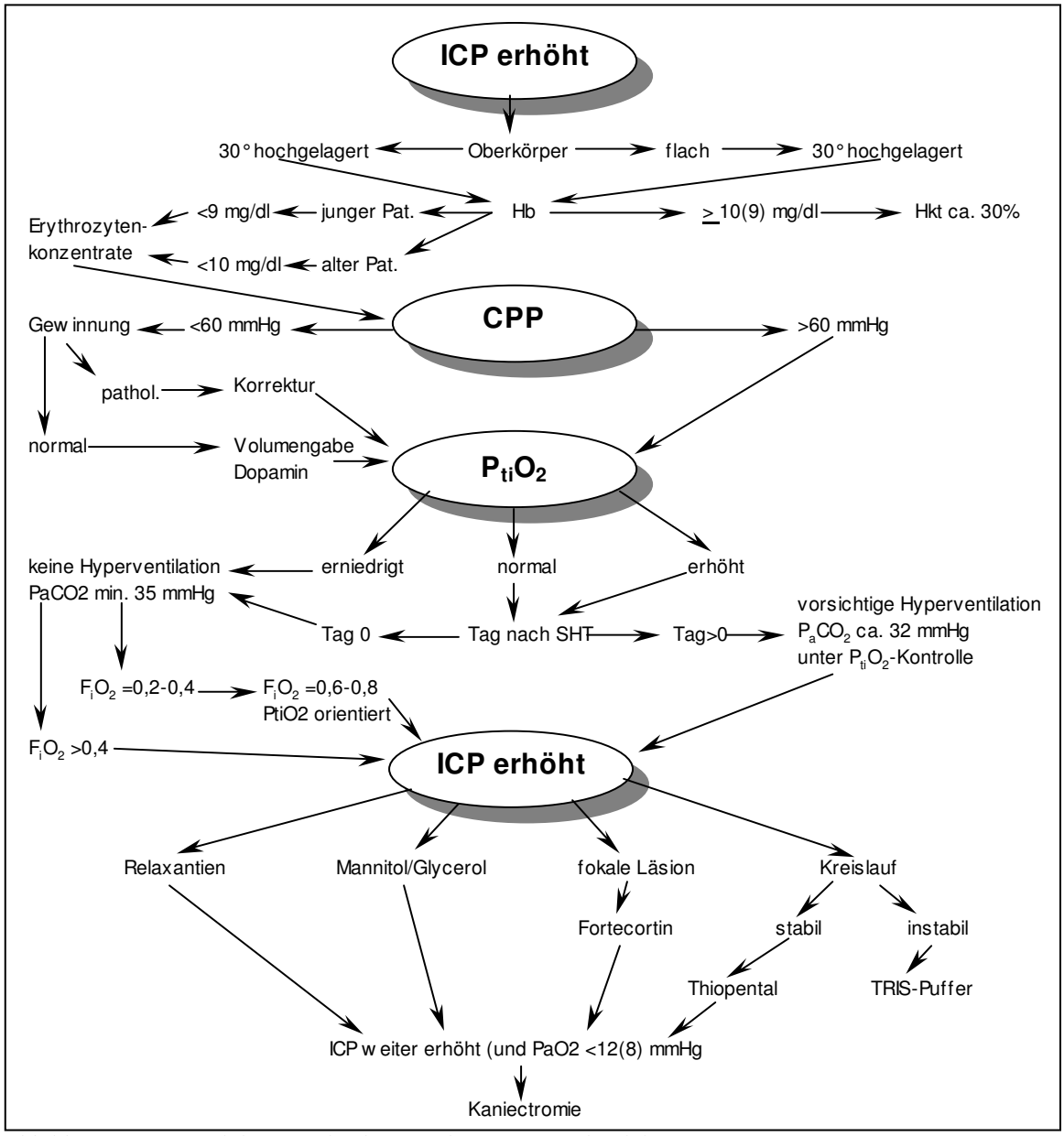


Abbildung 8.4 Beispiel der Entscheidungsfindung zur Hirndrucktherapie

9 Problemstellung und Modellierung

9.1 Problemstellung

Ziel der Untersuchung ist es, aus den Daten, die in Kapitel 8 beschrieben wurden, einen Teilbereich zu extrahieren und diese Messwerte in allen möglichen Kombinationen zu analysieren.

Die Bereiche der evozierten Potentiale und der Glasgow Outcome Scores liegen in einem normierten Wertebereich von 1 bis 5. Diese Skalen werden je Bereich für drei Attribute angewandt.

Die Daten der Läsionen (Art, Lage und Volumen) werden innerhalb der Lage und der Art referentiell hinterlegt. Die Abbildung 9.1 veranschaulicht die ursprünglichen Fakten und Zusammenhänge der Lokalisationen jedes einzelnen Elipsoids.

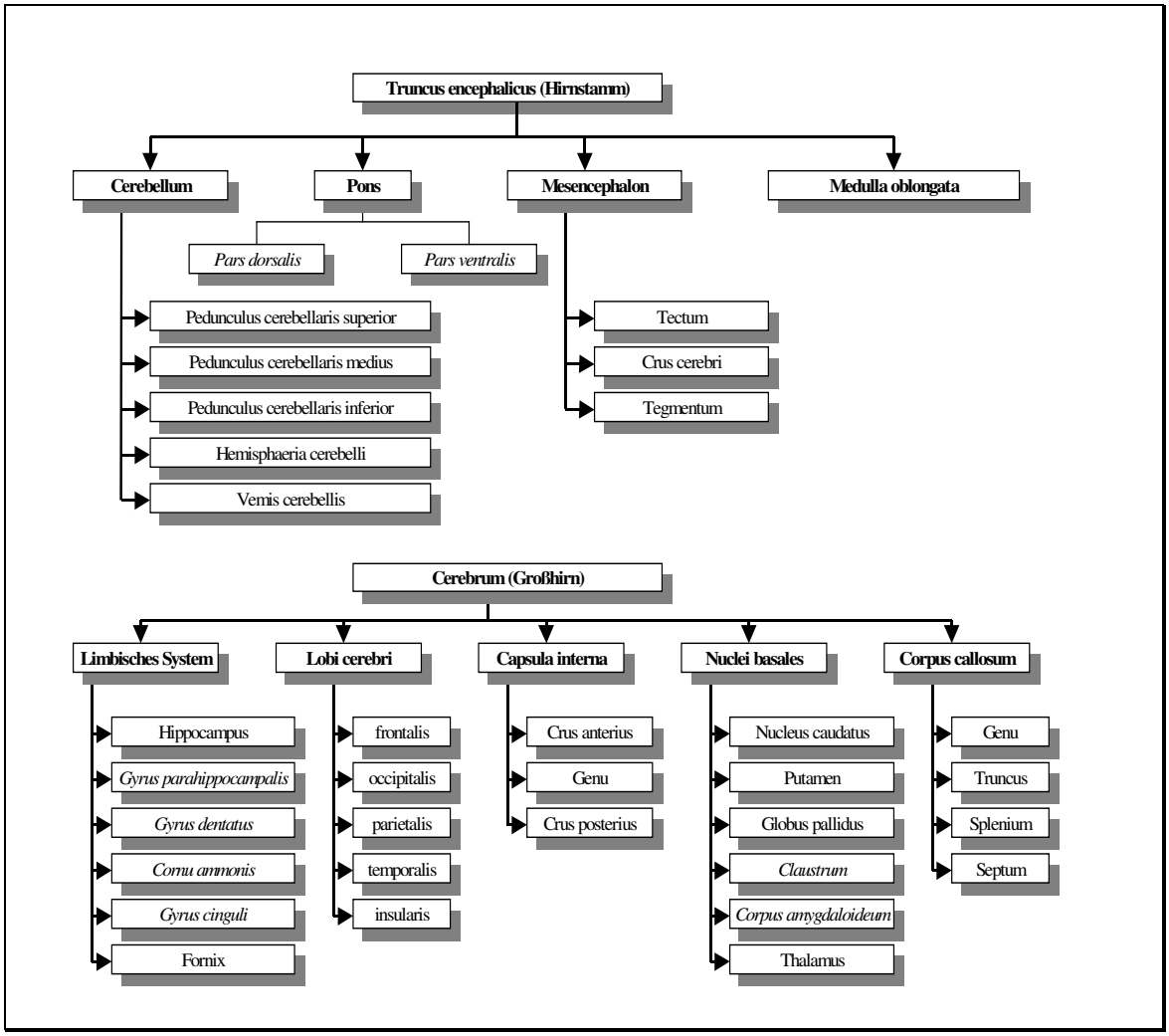


Abbildung 9.1 Beschreibung und Zusammenhänge der Lage einer Läsion

Die innerhalb der Analyse zu betrachtenden Daten werden anhand der Abbildung 9.2 für Lage und Abbildung 9.3 für Art einer Läsion transformiert.

Durch die Reduzierung des ursprünglichen Wertebereichs gehen einige Daten und Ausprägungen verloren. So reduziert sich in dieser Arbeit die Anzahl der Läsionsdaten von 473 auf 320 Datensätze.

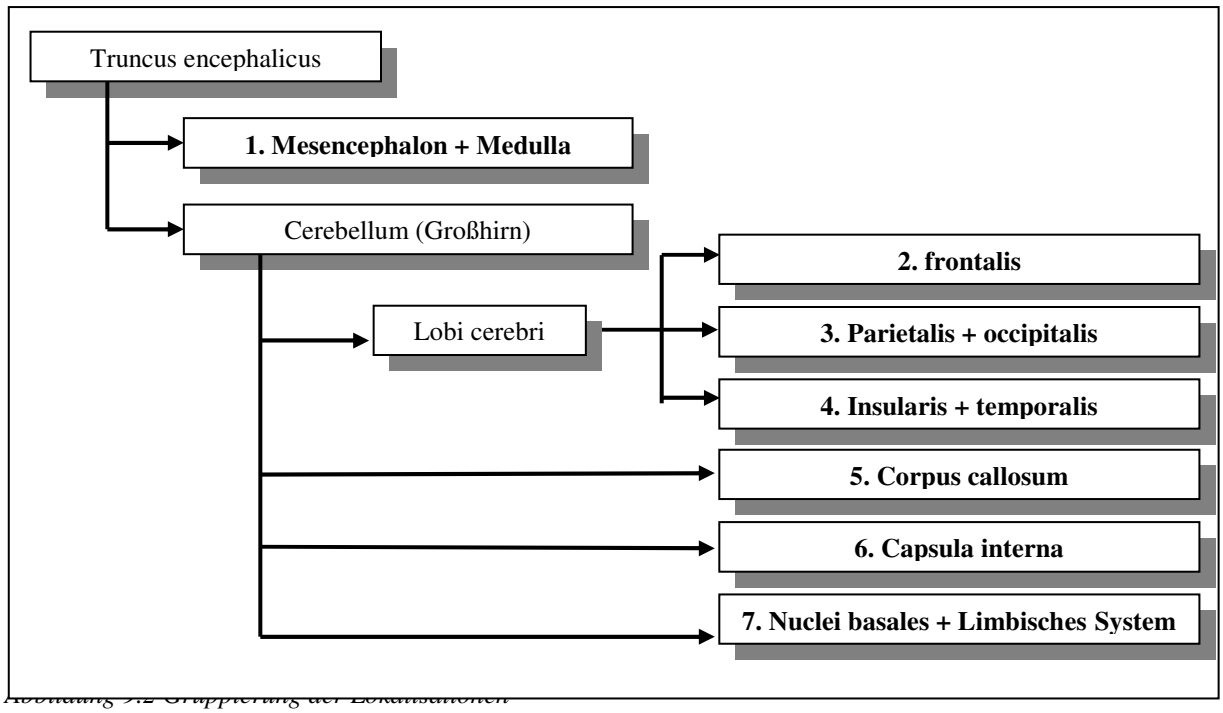
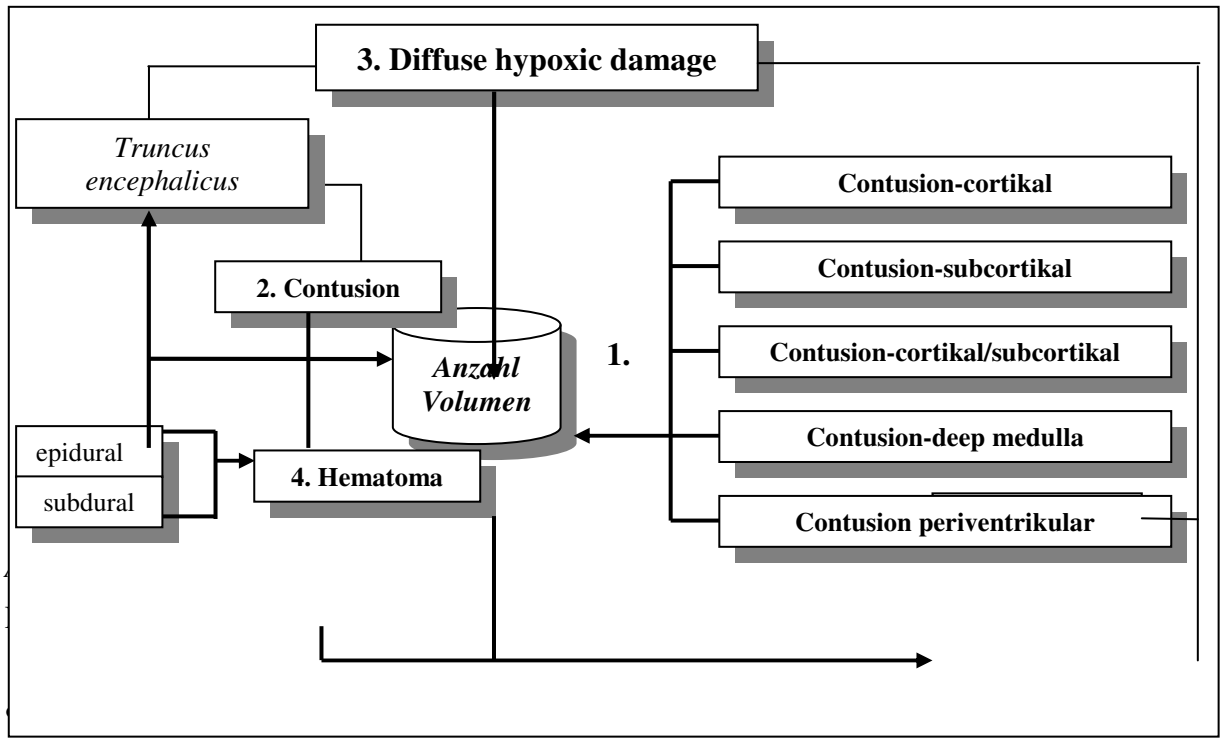


Abbildung 9.2 Gruppierung der Lokalisationen



Diese Werte werden in den Klassen Akustik, Medianus, Tibialis ermittelt und innerhalb der Skala [1;5] abgelegt.

- Glasgow Outcome Score (GOS):

Diese Daten stehen ebenfalls in einer Skala von [1;5] zur Verfügung und werden nach Entlassung, 3 Monaten und 12 Monaten berechnet.

- Läsionslage:
  Die Beschreibung der Lokalisationen wird als Primärschlüssel angewandt und steht in sieben Ausprägungen zur Verfügung. Die Daten sind nahezu gleichverteilt.

- Läsionsarten:
  Die medizinische Beschreibung der Fakten dient als Primärschlüssel der Referenzen und sind in vier Ausprägungen integriert. Die Läsion "Diffuse hypoxic damage" ist nur bei einer Läsion gespeichert, wodurch eine Analyse auf diesen Sektor irrelevant ist.

Anhand dieser Beschreibung lassen sich drei Gruppen der Daten definieren:

- *Gruppe I: Verletzung*
  Die Daten der Verletzungen (Art, Lage und Volumen) werden innerhalb dieser 1. Gruppe integriert, d.h. jede Läsion wird anhand der beschreibenden Daten klassifiziert.

- *Gruppe II: Evozierte Potentiale*
  Die Messwerte der evozierten Potentiale, ermittelt im Bereich der Akustik, des Medianus und des Tibialis, werden in dieser Gruppe zusammengefasst. Die Daten werden somit je Patient ermittelt und abgelegt.

- *Gruppe III: Glasgow Outcome Score*
  Die Prognosewerte, die mit den Eigenschaften nach Entlassung, 3 Monaten und 12 Monaten ermittelt werden, können in dieser Gruppe zusammengefasst werden und beschreiben den Zustand des Patienten zum jeweiligen Zeitpunkt.

Durch diese Gruppendefinitionen lassen sich folgende Abhängigkeiten vermuten, die in den sich anschließenden Kapiteln analysiert werden.

- *Gruppe III von Gruppe I*:
  Dieser Zusammenhang prognostiziert das Krankheitsbild eines Patienten anhand der zur Verfügung stehenden Daten der Verletzungen im Bereich der Neurologie.
  Kann eine Prognose anhand der Läsionsdaten aus Gruppe I berechnet werden?

- *Gruppe II von Gruppe I*:

  Die Abhängigkeit der Messwerte der evozierten Potentiale bzgl. der Verletzungsdaten ermöglicht den Rückschluss auf die Notwendigkeit des Messens einzelner Bereiche der EP-Werte.

  Besteht die Möglichkeit, aufgrund der Daten einer Verletzung auf die EP-Werte zu schließen, um dadurch das Ermitteln einzelner EP-Bereiche zu erübrigen?

- *Gruppe III von Gruppe II:*

  In dieser Analyse der Daten wird der Zusammenhang zwischen den Prognosewerten des Glasgow Outcome Scores und den Messwerten der evozierten Potentiale untersucht.

  Kann aufgrund der evozierten Potentiale eines Patienten auf die Glasgow Outcome Scores zu den drei Zeitpunkten eine Zukunftsprognose erzeugt werden?

- *Gruppe II von Gruppe III:*

  Der Zusammenhang der Prognosewerte des Glasgow Outcome Scores bzgl. der Messwerte des evozierten Potentials ermöglicht Rückschlüsse auf die Potentiale und somit die Reduzierung der Messreihen.

  Kann durch die Existenz der Glasgow Outcome Scores auf die evozierten Potential zurückgeschlossen bzw. können diese überprüft werden?

- *Gruppe II (III) von Gruppe II (III)*

  Mit Hilfe der Analyse dieser Zusammenhänge ist es möglich die Korrelationen innerhalb der Gruppen „Evozierte Potentiale" bzw. „Glasgow Outcome Score" herauszustellen.

  Wie stark sind die inneren Zusammenhänge der EP-Werte und der Prognosewerte des GOS?

In den sich anschließenden Kapiteln werden diese Daten relational strukturiert und in allen Abhängigkeiten statistisch und neuronal aufbereitet. In Abbildung 9.4 werden die analysierten Abhängigkeiten graphisch dargestellt.

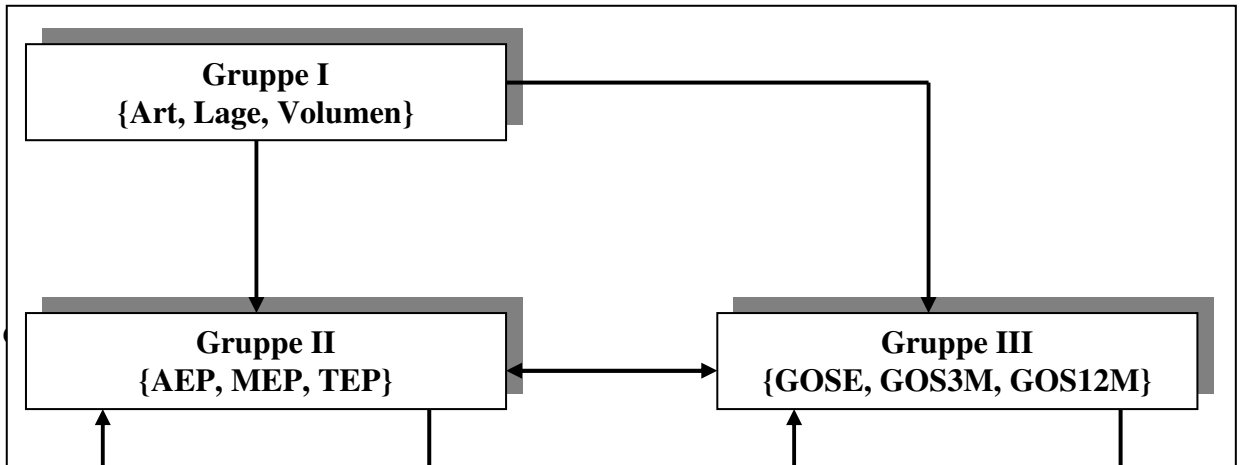


Abbildung 9.4 Übersicht der zu untersuchenden Abhängigkeiten

Im Kapitel 10 werden die beschriebenen Abhängigkeiten mittels des Stuttgarter Simulators „SNNS 4.1" untersucht und interpretiert.
Die Zusammenhänge der Gruppen werden anschließend anhand der einfachen, zweifachen und multiplen Regressions-/Korrelationsanalyse untersucht (Kapitel 11).
Zusätzlich wird durch eine Filtrierung auf die Referenzen Art bzw. Lage eine Art-/Lagespezifikation durchgeführt und die entsprechenden Korrelationskoeffizienten ermittelt.

Durch die beliebigen Kombinationsmöglichkeiten der Input- und Outputwerte und der zugehörigen Transformationsfunktionen bietet das entwickelte Tool „KNN 1.0" die Funktionen, die benötigt werden, um alle Variationen statistisch aufbereiten und analysieren zu können.

9.2 Die Datenbank

Bei der vom Städtischen Klinikum Fulda gelieferten Datenbank handelt es sich um eine unter „Microsoft Access 97“ entwickelte Datenbank, die die Daten in anonymisierter Form beinhaltet. Die Datenbankstruktur wird in Abbildung 9.5 dargestellt, wobei die Relationen der Tabellen an den direkten Verbindungen abzulesen sind. Aufgrund der großen Attributmengen der Tabellen „Identifikation“, „Läsion“ und „EP“ ist die Darstellung auf die für diese Arbeit nötigen Felder beschränkt.

Die Datensätze beschreiben die Messwerte der evozierten Potentiale

(EP = {AEP, ScoreEPMed, ScoreEPTib}) von 30 Patienten und deren 473 Läsionen

(Läsion = {Läsionstyp, Lokalisation, Volumen})

Die Daten der Tabelle „Identifikation“ werden unter Berücksichtigung der eindeutigen Patientennummer (PATNr) über eine 1:n-Verbindung mit den Läsions-Daten und mit einer 1:1-Verbindung mit den evozierten Potentialen (EP) verbunden.

Die beiden Referenztabellen (1:1-Verbindung) „DB Lesionstyp“ und „DB_Lesionsloc“ werden über die Schlüsselfelder „Läsionstyp“ bzw. „Lokalisation“ mit der Primärtabelle „Läsion“ verknüpft.

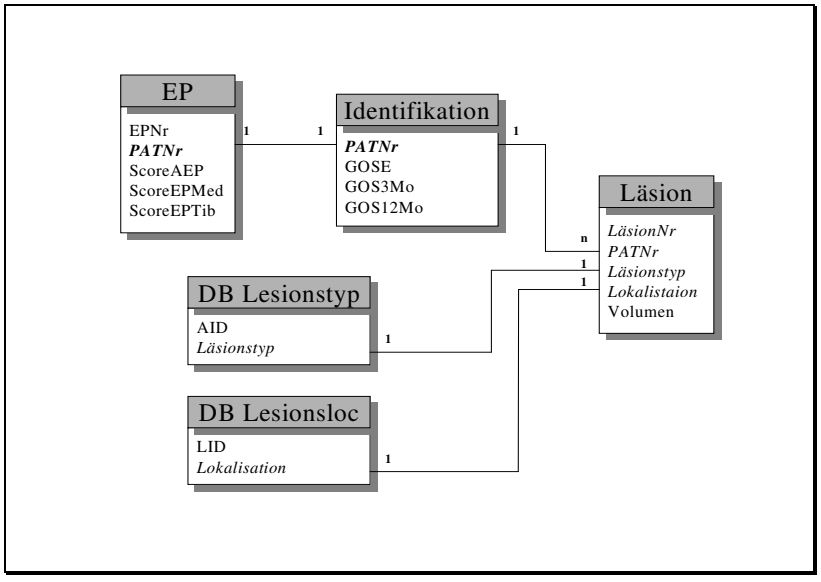


Abbildung 9.5 Struktur der Datenbank des Städtischen Klinikums Fulda

9.3 Die anonymisierten Daten

Die zur Verfügung gestellten Daten lassen sich in zwei Bereiche unterteilen. Der erste Bereich wird durch die Patientendaten beschrieben, der zweite durch die Läsionsdaten.
Die Datensätze befinden sich wie zuvor beschrieben in den Tabellen „Identifikation“, „EP“ und „Läsion“.

- Patientendaten:

  Es handelt sich hierbei um anonymisierte Daten, die neben einem Schlüsselfeld die Werte des evozierten Potentials und der Glasgow Outcome Scores enthalten.
  Die evozierten Potentiale (EP) erstrecken sich über einen Wertebereich von {1; 2; 3; 4; 5} und werden in den Kategorien akustischer EP (ScoreAEP), Medianus EP (ScoreEPMed) und Tibialis EP (ScoreEPTib) gemessen und abgelegt.
  Der Glasgow Outcome Score erstreckt sich in der Ursprungsdatenbank ebenfalls über einen Wertebereich von {1; 2; 3; 4; 5} und wird je Patient bei Entlassung (GOSE), sowie 3 Monate (GOS3M) und 12 Monate später (GOS12M) ermittelt.
  Mit Hilfe dieser Skalenwerte ist es möglich das Krankheitsbild eines Patienten zu diagnostizieren.

- Läsionsdaten:

  Die verschiedenen Läsionen setzen sich aus den referentiell unterstützten Feldern der Tabellen „DB-Lesionloc“ mit 65 verschiedenen Lokalisationen und den Feldern aus „DB-Lesionstyp“ mit 28 Läsionstypen zusammen. Zusätzlich zu den Informationen über Art und Lage stehen die Daten bzgl. der Länge, Breite und Tiefe des Elipsoids und somit auch das ableitbare Volumen ($V=4\pi abc/3$) zur Verfügung.
  Die verschiedenen Unterklassifizierungen wie z.B. Ödem (Ja/Nein) oder MRT_x (True/False) interessieren in diesem Forschungsprojekt nicht und werden daher vernachlässigt.

9.4 Die Datenübernahme

Aufgrund der Problemstellung für diese Diplomarbeit wird die Struktur der Datenbank entsprechend angepasst. Diese Anpassung im Bereich der Datenstruktur kann aus Abbildung 9.6 entnommen werden. Die Beziehungen der Tabellen sind an den Verbindungen ablesbar.

Die Felder „Lokalisation“ und „Läsionstyp“ innerhalb der Läsionstabelle werden durch die indizierten Felder LID und AID der Referenztabellen „Lage“ und „Art“ ersetzt, wodurch die referentielle Integrität gewahrt werden kann.

Die beiden Tabellen EP und Identifikation der Datenbank des Städtischen Klinikums Fulda werden zu einer neuen Tabelle „Patient“ zusammengeführt, die alle Datensätze beider Tabellen enthält. Die eindeutige Zuordnung wird über das Schlüsselfeld „Patientennummer“ (PatNr bzw. PID) gewährleistet.

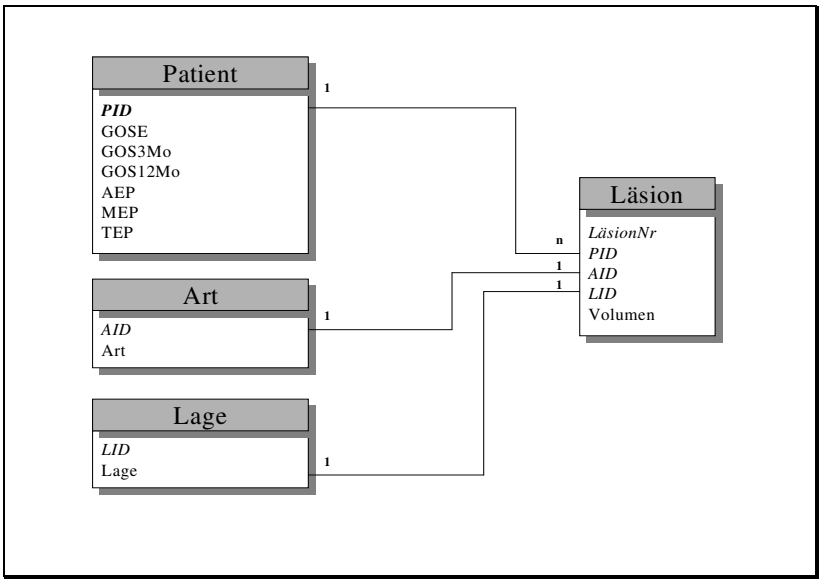


Abbildung 9.6 Struktur der Datenbank von KNN

9.5 Die Datenkonvertierung

Für das Projekt „Künstliche neuronale Netze in der Neurochirurgie“ müssen diese Daten so aufbereitet werden, dass nur die relevanten Felder herausgezogen werden (Reduzierung der Daten).
Anschließend werden diese gefilterten Informationen so normalisiert, dass eine referentielle Integrität gewährleistet werden kann.
Durch diese Konvertierung ist es möglich eine Eindeutigkeit innerhalb der Zuweisungen von Lage und Art zu den verschiedensten Läsionstypen in Kombination mit den Lokalisationen innerhalb der Datenbank KNN zu definieren.

Aus den zuvor beschriebenen Daten des Klinikums und der für diese Arbeit benötigten Informationen ergibt sich die Konvertierungsproblematik, die aus Tabelle 9.1 zu entnehmen ist.

Pos.	Tabelle	Attribut	Typ	Wertebereich
1.	**Patient**	PID	Zähler	[1;30]
2.		GOSE	Integer	[2;5]
3.		GOS3M	Integer	[2;5]
4.		GOS12M	Integer	[2;5]
5.		AEP	Integer	[1;4]
6.		MSEP	Integer	[1;4]
7.		TSEP	Integer	[1;4]
8.	**Läsion**	LäsionsID	Zähler	[1;320]
9.		PID	Integer	[1;30]
10.		LID	Integer	[1;7]
11.		AID	Integer	[1;4]
12.		VOL	Long Integer	[0;189]

Tabelle 9.1 Klassifikation der relevanten Daten

Bei den als Zähler definierten Feldern handelt es sich um Indizes, die fortlaufend erhöht werden, um eine eindeutige Beziehung innerhalb der Datenbank gewährleisten zu können.
Die mit dem Typ „Integer“ versehenen Felder entsprechen einer Skala natürlicher Zahlen innerhalb des Wertebereichs, wobei nach Definition der Wertebereich der Glasgow Outcome-Werte auf ein Intervall der natürlichen Zahlen von [1;3] zu beschränken ist. Dabei werden die Werte {1; 2}-1, {3}-2 und {4; 5}-3 zugewiesen.

Der ableitbare Wert VOL beschreibt das Volumen der charakterisierten Läsion und nimmt somit eine Zahl im Wertebereich von „long integer“ an. Diese Messwerte ergeben sich aus den Ausprägungen der Läsionen, gemessen in Millimeter und der Volumenformel (3,14159*[Größe-re-li]*[Größe-an-po]*[Größe-cr-ca])/6000, gemessen in ml.

Die Tabelle „Patient“ setzt sich aus den beiden Tabellen „EP“ und „Identifikation“ der ursprünglichen Datenbank des Städtischen Klinikums Fulda zusammen, während die Tabelle „Läsion“ aus den Tabellen „DB-Lesionloc“, „DB-lesiontyp“ und „Läsion“ der Ursprungsdatenbank entsteht.

Die Tabellen des Städtischen Klinikums werden über einen Datenfilter reduziert und normalisiert. Die strukturellen Änderungen veranschaulicht Abbildung 9.7.

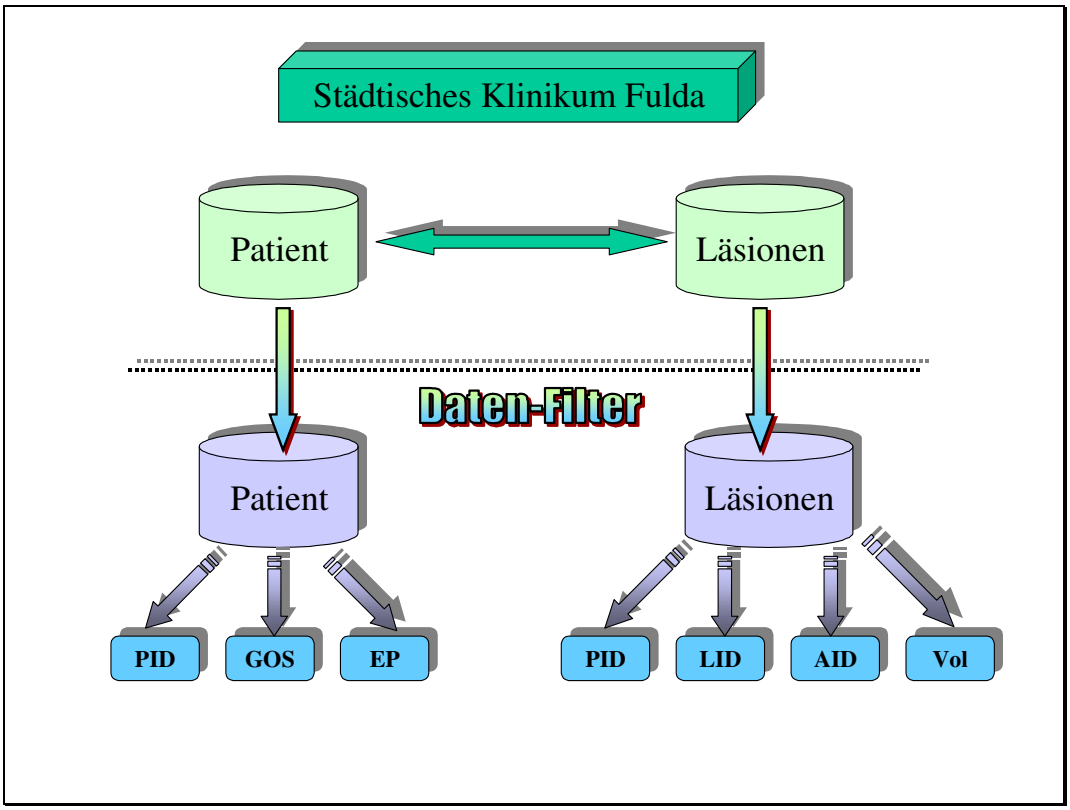


Abbildung 9.7 Datenkonvertierung Städtisches Klinikum Fulda

9.6 Die Datentransformation

Diese Transformation lässt sich in zwei Bereiche gliedern:

1. Patientendaten:

   Die Daten der evozierten Potentiale und des Glasgow Outcome Scores werden mit Hilfe der „Patienten-Nr“ zu einer neuen Tabelle „Patient“ unter dem Schlüssel „PID“ zusammengeführt.
   Innerhalb dieser Transformation werden die Daten des Glasgow Outcome Scores in den neuen Wertebereich von {1; 2 ;3} überführt. Der neue GOS-Wert von 1 wird durch den alten Wert von 1 oder 2 erreicht, der neue Wert von 2 durch 2 und der neue Wert von 3 durch die alten Werte von 4 oder 5 ersetzt.
   Der Datentyp dieser Felder wird von ursprünglich „Text“ auf „Integer“ gesetzt, um somit innerhalb der Berechnungen auf die konkreten Werte zugreifen zu können.

2. Läsionsdaten:

   Die Daten der Lokalisation und der Läsionstypen werden mit Hilfe der Referenztabellen für Art und Lage verschlüsselt.
   Dabei wird der Wertebereich der Lokalisationen von ursprünglich 65 Ausprägungen zunächst auf 14 Gruppen/Klassen beschränkt und anschließend auf die 7 neuen Entitäten (Tabelle 9.2) gebündelt. Durch diese Transformation wird der Datenbestand um 90 Datensätze auf 383 reduziert.

LID	Location
1	**Mesencephalon Pons Medulla**
2	**Frontalis**
3	**Parietalis occipitalis**
4	**Insularis temporalis**
5	**Corpus callosum Septum**
6	**Capsula interna**
7	**Nuclei basales Limbisches System**

Tabelle 9.2 Referenztabelle Lage

Der Wertebereich der Läsionstypen von ursprünglich 28 Ausprägungen wird auf 14 Gruppen/Klassen reduziert und anschließend auf 4 neue Entitäten (Tabelle 9.3) abgebildet. Innerhalb dieser Konvertierung werden die Daten um 63 Datensätze auf 320 vermindert.

AID	Art
1	**Diffuse hypoxic damage**
2	**Hematoma**
3	**Contusion Cerebrum**
4	**Contusion Truncus**

Tabelle 9.3 Referenztabelle Art

9.7 Die Datenbankstruktur von KNN 1.0

Nach dieser Transformation stehen der Analyse die 30 Patientendaten incl. der konvertierten GOS-Werte und der 320 Läsiondaten, deren Reduzierung auf die Skalentransformation der Referenzen „Lage“ und „Art“ zurückzuführen ist, zur Verfügung.
Die Struktur der Datenbank ist in Abbildung 9.5 dargestellt.
Es stehen zwei Datentabellen zur Verfügung. Diese sind „Patient“ und „Läsion“, wobei es sich hierbei um eine 1:n-Verknüpfung handelt. Die referentielle Integrität wird im Bereich des Patienten durch den Primärschlüssel PID und in der Tabelle „Läsion“ durch den Fremdschlüssel PID mit Primärschlüssel LäsionsID gewährleistet.

Die beiden Referenztabellen „Art“ und „Lage“ komplettieren die relationale Datenbank durch den jeweiligen Index in Verbindung mit der Läsionstabelle. Bei diesen Verbindungen handelt es sich um eine 1:1-Beziehung, die den Zugriff auf die reduzierte Lage- und Art-Entität zulässt.

Entitäten:

- Patient = {*PID*, GOSE, GOS3M, GOS12M, AEP, MSEP, TSEP}
- Läsion = {*LäsionsID*, PID, LID, AID, VOL}
- Lage = {*LID*, Beschreibung}
- Art = {*AID*, Beschreibung}

Die Beschreibung der einzelnen Felder und deren Attribute kann man Tabelle 9.4 entnehmen.
Die Ausprägungen der einzelnen Messwerte und Skalen ergeben sich aus den in Tabelle 9.1 beschriebenen Wertebereichen.

Pos.	Attribut	Beschreibung

1.	PID	Patienten Identifikations-Nummer
2.	GOSE	Glasgow Outcome Score bei Entlassung
3.	GOS3M	Glasgow Outcome Score nach 3 Monaten
4.	GOS12M	Glasgow Outcome Score nach 12 Monaten
5.	AEP	Akustisches evoziertes Potential
6.	MSEP	Medialis evoziertes Potential
7.	TSEP	Tibialis evoziertes Potential
8.	LäsionsID	Läsions Identifikations-Nummer
9.	LID	Läsions ID als Fremdschlüssel zu Lage
10.	AID	Art ID als Fremdschlüssel zu Art
11.	VOL	Berechnetes Volumen in Milliliter (ml)

Tabelle 9.4 Eigenschaften der Entitäten von KNN 1.0

10 Analyse mittels SNNS 4.1

10.1 Einleitung

Die Klinikdaten werden in reduzierter Form in verschiedene künstliche neuronale Netze übergeben. Drei Netze werden hier näher erläutert, es handelt sich um feedforward strukturierte Topologien. Durch differenzierte Dateinamen lässt sich die verwendete Struktur erkennen.
Da der Stuttgarter Simulator in der Windows-Version nur Werte zwischen Null und Eins liefert, müssen der Eingabe- und der Ausgaberaum normalisiert werden bzw. über entsprechende Transformationsfunktionen den Räumen angepasst werden
Die gelieferten Ergebnisse aus den Resultfiles werden analysiert und bewertet. Die Werte der Varianzen, Standardabweichungen und die des mittleren quadratischen Fehlers liegen im Tool KNN 1.0 in der Tabelle Auswertung und stehen über die Benutzungsoberfläche des Programms zur Verfügung.
Die Daten, die als Berechnungsgrundlage dienen, werden im folgenden näher erläutert.

10.2 Die Architektur der verwendeten Netze

Enthält der Dateiname eines Ergebnisfiles die Zeichenfolge „32-5“, so kann man daraus auf die Architektur des verwendeten KNN schließen. Das Netz hat somit die Feedforward-Struktur: (FFW 32-5)

Input-neuronen	Hidden Layer1	Hidden Layer2	Hidden Layer 4	Hidden Layer5	Output-neuronen
3	32	16	8	5	3

Tabelle 10.1 Architektur I Feedforward-Netz

Enthält das Ergebnisfile die Zeichenfolge „320“, so ist das künstliche neuronale Netz mächtiger und hat die Form:

Input-neuronen	Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5	Hidden 6	Hidden 7	Output-neuronen
3	320	160	80	40	20	10	5	3

Tabelle 10.2 Architektur II Feedforward Netz

Die Result-Files mit der Bezeichnung „el_*“ enthalten die Ergebnisdaten, die von sogenannten Elman-Netzen produziert werden. Die Elman-Netze liegen in folgender Form vor:

Input neuronen	Hidden 1	Hidden 2	Hidden 3	Output neuronen
3	27	18	9	3

Tabelle 10.3 Architektur Elman Netz

10.3 Die Trainingszyklen

Die 32-5 Feedforward-Netze und Elman-Netze wurden u.a. in 500 Trainingszyklen trainiert.

Die 320er Feedforward-Netze werden für [PatientenDaten – Glasgow Outcome Score] und [Glasgow Outcome Score – Evozierte Potentiale] mit je linearer Input- und Outputfunktion mit 100 Zyklen trainiert.

Für [Läsionsdaten – Evozierte Potentiale] und je linearer Input- und Outputfunktion werden nur 10 Trainingszyklen durchgeführt.

10.4 Beschreibung der trainierten Daten im SNNS

Im folgenden Abschnitt werden zunächst die an den erzeugten neuronalen Netzen eingehenden Inputdaten sowie die Outputdaten aufgezeigt, mit welchen die Netze trainiert werden. Die jeweiligen Abhängigkeiten in den Patternfiles bzw. Ergebnisfiles lassen sich aus Tabelle 10.12 entnehmen.

Es werden vier mögliche Prognosen untersucht, und zwar:

1. *Das Netz soll versuchen, anhand der Läsionsdaten den Glasgow Outcome Score zu bestimmen.*
2. *Das Netz soll versuchen, anhand der Läsionsdaten die evozierten Potentiale zu bestimmen.*
3. *Das Netz soll versuchen, anhand des Glasgow Outcome Scores die evozierten Potentiale zu bestimmen.*
4. *Das Netz soll versuchen, anhand der evozierten Potentiale den Glasgow Outcome Score zu bestimmen.*

Darunter befinden sich jeweils die Varianzen, Standardabweichungen und der mittlere quadratische Fehler der vom SNNS gelieferten Ergebnisse, die in der Tabelle „Auswertung“ im Tool KNN 1.0 (siehe auch Kapitel 12) zur Verfügung gestellt sind.

Abhängigkeit: Läsionsdaten (Input) – Glasgow Outcome Score (Output)

INPUT1	Art der Läsion	}	SNNS Netz	{	GOSE OUTPUT1
INPUT2	Lage der Läsion	}	SNNS Netz	{	GOS3M OUTPUT2
INPUT3	Volumen der Läsion	}	SNNS Netz	{	GOS12M OUTPUT3

In der Auswertung werden, wie eingangs aufgezeigt, die Sollwerte der jeweiligen Datenbank mit den durch den Simulator berechneten Werten verglichen. So ergeben sich folgende Ergebnisse:

Varianz1	bezüglich	GOSE	soll	und	GOSE	berechnet

Varianz2	bezüglich	GOS3M	soll	und	GOS3M	berechnet
Varianz3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

Standardabw.1	bezüglich	GOSE	soll	und	GOSE	berechnet
Standardabw.2	bezüglich	GOS3M	soll	und	GOS3M	berechnet
Standardabw.3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

Fehler² 1	bezüglich	GOSE	soll	und	GOSE	berechnet
Fehler² 2	bezüglich	GOS3M	soll	und	GOS3M	berechnet
Fehler² 3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

Abhängigkeit: Läsionsdaten (Input) – Evozierte Potentiale (Output)

INPUT1	Art der Läsion	} SNNS Netz {	AEP	OUTPUT1
INPUT2	Lage der Läsion		MSEP	OUTPUT2
INPUT3	Volumen der Läsion		TSEP	OUTPUT3

In der Auswertung werden auch hier die Sollwerte mit den durch den Simulator berechneten Werten verglichen. So ergeben sich folgende Ergebnisse:

Varianz1	bezüglich	AEP	soll	und	AEP	berechnet
Varianz2	bezüglich	MSEP	soll	und	MSEP	berechnet
Varianz3	bezüglich	TSEP	soll	und	TSEP	berechnet

Standardabw.1	bezüglich	AEP	soll	und	AEP	berechnet
Standardabw.2	bezüglich	MSEP	soll	und	MSEP	berechnet
Standardabw.3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

Fehler² 1	bezüglich	AEP	soll	und	AEP	berechnet
Fehler² 2	bezüglich	MSEP	soll	und	MSEP	berechnet
Fehler² 3	bezüglich	TSEP	soll	und	TSEP	berechnet

Abhängigkeit: Glasgow Outcome Score (Input) – Evozierte Potentiale (Output)

INPUT1	GOSE		AEP	OUTPUT1
INPUT2	GOS3M	SNNS Netz	MSEP	OUTPUT2
INPUT3	GOS12M		TSEP	OUTPUT3

In der Auswertung werden auch hier die Sollwerte mit den durch den Simulator berechneten Werten verglichen. So ergeben sich folgende Ergebnisse:

Varianz1	bezüglich	AEP	soll	und	AEP	berechnet
Varianz2	bezüglich	MSEP	soll	und	MSEP	berechnet
Varianz3	bezüglich	TSEP	soll	und	TSEP	berechnet

Standardabw.1	bezüglich	AEP	soll	und	AEP	berechnet
Standardabw.2	bezüglich	MSEP	soll	und	MSEP	berechnet
Standardabw.3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

Fehler² 1	bezüglich	AEP	soll	und	AEP	berechnet
Fehler² 2	bezüglich	MSEP	soll	und	MSEP	berechnet
Fehler² 3	bezüglich	TSEP	soll	und	TSEP	berechnet

Abhängigkeit: Evozierte Potentiale (Input) - Glasgow Outcome Score (Output)

INPUT1	AEP	} SNNS Netz {	GOSE	OUTPUT1
INPUT2	MEP		GOS3M	OUTPUT2
INPUT3	TEP		GOS12M	OUTPUT3

In der Auswertung werden auch hier die Sollwerte mit den durch den Simulator berechneten Werten verglichen. So ergeben sich folgende Ergebnisse:

Varianz1	bezüglich	GOSE	soll	und	GOSE	berechnet
Varianz2	bezüglich	GOS3M	soll	und	GOS3M	berechnet
Varianz3	bezüglich	GOS12M	soll	und	GOS12M	berechnet
Standardabw.1	bezüglich	GOSE	soll	und	GOSE	berechnet
Standardabw.2	bezüglich	GOS3M	soll	und	GOS3M	berechnet
Standardabw.3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

Fehler² 1	bezüglich	GOSE	soll	und	GOSE	berechnet
Fehler² 2	bezüglich	GOS3M	soll	und	GOS3M	berechnet
Fehler² 3	bezüglich	GOS12M	soll	und	GOS12M	berechnet

10.5 Die Ergebnisse des Stuttgarter Simulators

Exemplarisch wird einerseits das beste, andererseits das schlechteste Ergebnis hinsichtlich der Outputwerte dargestellt. Zusätzlich wird Auskunft über die Transformationsfunktionen und das dazugehörende Netz gegeben. Als Beispiel wird die Varianz, berechnet aus den Sollwerten und den berechneten Werten des Netzes, angegeben.

Abhängigkeit: GOSy = f (LÄSION)

Inputdaten sind die Läsionsdaten (320 Datensätze standen zur Verfügung), die somit wie oben beschrieben die Art der Läsion (Input 1), die Lage der Läsion (Input 2) und das Volumen der Läsion (Input 3) enthalten.
Als Ergebnis wird der Glasgow Outcome Score bei Entlassung (Output 1), nach drei Monaten (Output 2) und nach 12 Monaten (Output 3) erwartet.

Tabellarische Aufstellung des besten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
GOSE	Logarithmisch	Linear	2	Elman	0,00449
GOS3M	Logarithmisch	Linear	2	Elman	0,007076
GOS12M	Logarithmisch	Linear	2	Elman	0,007051

Tabelle 10.4 Beste SNNS Ergebnisse GOSy = f (LÄSION)

Tabellarische Aufstellung des ungeeignetsten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
GOSE	Kehrwert	Logarithmisch	18	FFW 32-5	0,175

GOS3M	Logarithmisch	Logarithmisch	12	FFW 32-5	0,267
GOS12M	Logarithmisch	Logarithmisch	12	FFW 32-5	0,268

Tabelle 10.5 Schlechteste SNNS Ergebnisse GOSy = f (LÄSION)

Abhängigkeit: yEP = f (LÄSION)

Als Eingangsdaten liegen die Läsionsdaten zu Grunde. Der erwartete Output soll die evozierten Potentiale liefern (Output 1 = AEP, Output 2 = MSEP, Output 3 = TSEP). Zum Training standen 320 Datensätze zur Verfügung.

Tabellarische Aufstellung des besten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
AEP	Exponentiell	Tangens Hyp.	40	FFW 32-5	0,001171
MSEP	Tangens Hyp.	Tangens Hyp.	50	FFW 32-5	0,006367
TSEP	Logarithmisch	Linear	34	FFW 32-5	0,008817

Tabelle 10.6 Beste SNNS Ergebnisse yEP = f (LÄSION)

Tabellarische Aufstellung des ungeeignetsten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
AEP	Logarithmisch	Logarithmisch	31	FFW 32-5	0,1055
MSEP	Kehrwert	Logarithmisch	41	FFW 32-5	0,2016
TSEP	Kehrwert	Logarithmisch	41	FFW 32-5	0,2695

Tabelle 10.7 Schlechteste SNNS Ergebnisse yEP = f (LÄSION)

Abhängigkeit: yEP = f (GOSx)

Die Inputwerte sind hier der Glasgow Outcome Score bei Entlassung, nach drei und nach zwölf Monaten, zum Training standen hier 30 Datensätze der Patienten zur Verfügung.

Tabellarische Aufstellung des besten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
AEP	Kehrwert	Tangens Hyp.	65	FFW 32-5	0,001875
MSEP	Kehrwert	Tangens Hyp.	65	FFW 32-5	0,002522
TSEP	Exponentiell	Tangens Hyp.	60	FFW 32-5	0,002897

Tabelle 10.8 Beste SNNS Ergebnisse yEP = f (GOSx)

Tabellarische Aufstellung des ungeeignetsten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
AEP	Linear	Logarithmisch	66	FFW 32-5	0,117732
MSEP	Tangens Hyp.	Logarithmisch	71	FFW 32-5	0,105909
TSEP	Tangens Hyp.	Logarithmisch	71	FFW 32-5	0,089151

Tabelle 10.9 Schlechteste SNNS Ergebnisse yEP = f (GOSx)

Abhängigkeit: GOSy = f (xEP)

Die Inputwerte sind hier die evozierten Potentiale (AEP, MEP, TEP), die Outputwerte stellen den Glasgow Outcome Score bei Entlassung (GOSE), nach drei Monaten (GOS3M) und nach zwölf Monaten (GOS12M) dar. Zum Training standen hier 30 Datensätze der Patienten zur Verfügung.

Tabellarische Aufstellung des besten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
GOSE	Kehrwert	TanH	95	FFW 32 – 5	0,000115

GOS3M	Kehrwert	TanH	95	FFW 32 – 5	0,000150
GOS12M	Kehrwert	TanH	95	FFW 32 – 5	0,000130

Tabelle 10.10 Beste SNNS Ergebnisse GOSy = f (xEP)

Tabellarische Aufstellung des ungeeignetsten Ergebnisses:

Output	Inputfunktion	Outputfunktion	Pattern-Nummer	Art des Netzes	Varianz
GOSE	Linear	Logarithmisch	76	FFW 32 – 5	0,049297
GOS3M	Linear	Logarithmisch	76	FFW 32 – 5	0,068100
GOS12M	Exponentiell	Logarithmisch	86	FFW 32 – 5	0,068099

Tabelle 10.11 Schlechteste SNNS Ergebnisse GOSy = f (xEP)

In Tabelle 10.12 befindet sich eine Übersichtsliste, in der beschrieben steht, welches Patternfile mit welchen Transformationsfunktionen bearbeitet wird, um die neuronalen Netze zu trainieren.

			IN			**OUT**		
Dateien	Abhäng.	Anzahl	Funktion	Divisor	Anzahl	Funktion	Divisor	Vol_Nor
1	LäsGosx	3	Linear	10	3	Linear	10	J
2	LäsGosx	3	Logarith		3	Linear	10	
3	LäsGosx	3	Expo		3	Linear	10	
4	LäsGosx	3	Kehrw		3	Linear	10	
5	LäsGosx	3	TanH		3	Linear	10	
6	LäsGosx	3	Linear	10	3	Logarith		J
7	LäsGosx	3	Linear	10	3	Expo		J
8	LäsGosx	3	Linear	10	3	Kehrw		J
9	LäsGosx	3	Linear	10	3	TanH		J
10	LäsGosx	3	Logarith		3	Expo		
11	LäsGosx	3	Logarith		3	Kehrw		
12	LäsGosx	3	Logarith		3	Logarith		
13	LäsGosx	3	Logarith		3	TanH		
14	LäsGosx	3	Expo		3	Logarith		
15	LäsGosx	3	Expo		3	Expo		
16	LäsGosx	3	Expo		3	Kehrw		
17	LäsGosx	3	Expo		3	TanH		
18	LäsGosx	3	Kehrw		3	Logarith		
19	LäsGosx	3	Kehrw		3	Expo		
20	LäsGosx	3	Kehrw		3	Kehrw		
21	LäsGosx	3	Kehrw		3	TanH		
22	LäsGosx	3	TanH		3	Logarith		
23	LäsGosx	3	TanH		3	Expo		
24	LäsGosx	3	TanH		3	Kehrw		
25	LäsGosx	3	TanH		3	TanH		
26	Läs_xEP	3	Linear	10	3	Logarith		J
27	Läs_xEP	3	Linear	10	3	Expo		J
28	Läs_xEP	3	Linear	10	3	Kehrw		J
29	Läs_xEP	3	Linear	10	3	Linear	10	J
30	Läs_xEP	3	Linear	10	3	TanH		J
31	Läs_xEP	3	Logarith		3	Logarith		
32	Läs_xEP	3	Logarith		3	Expo		
33	Läs_xEP	3	Logarith		3	Kehrw		
34	Läs_xEP	3	Logarith		3	Linear	10	
35	Läs_xEP	3	Logarith		3	TanH		
36	Läs_xEP	3	Expo		3	Logarith		
37	Läs_xEP	3	Expo		3	Expo		
38	Läs_xEP	3	Expo		3	Kehrw		
39	Läs_xEP	3	Expo		3	Linear	10	
			IN			**OUT**		
Dateien	Abhäng.	Anzahl	Funktion	Divisor	Anzahl	Funktion	Divisor	Vol_Nor

40	Läs_xEP	3	Expo		3	TanH		
41	Läs_xEP	3	Kehrw		3	Logarith		
42	Läs_xEP	3	Kehrw		3	Expo		
43	Läs_xEP	3	Kehrw		3	Kehrw		
44	Läs_xEP	3	Kehrw		3	Linear	10	
45	Läs_xEP	3	Kehrw		3	TanH		
46	Läs_xEP	3	TanH		3	Logarith		
47	Läs_xEP	3	TanH		3	Expo		
48	Läs_xEP	3	TanH		3	Kehrw		
49	Läs_xEP	3	TanH		3	Linear	10	
50	Läs_xEP	3	TanH		3	TanH		
51	GosxEP	3	Logarith		3	Logarith		
52	GosxEP	3	Logarith		3	Expo		
53	GosxEP	3	Logarith		3	Kehrw		
54	GosxEP	3	Logarith		3	Linear	10	
55	GosxEP	3	Logarith		3	TanH		
56	GosxEP	3	Expo		3	Logarith		
57	GosxEP	3	Expo		3	Expo		
58	GosxEP	3	Expo		3	Kehrw		
59	GosxEP	3	Expo		3	Linear	10	
60	GosxEP	3	Expo		3	TanH		
61	GosxEP	3	Kehrw		3	Logarith		
62	GosxEP	3	Kehrw		3	Expo		
63	GosxEP	3	Kehrw		3	Kehrw		
64	GosxEP	3	Kehrw		3	Linear	10	
65	GosxEP	3	Kehrw		3	TanH		
66	GosxEP	3	Linear	10	3	Logarith		J
67	GosxEP	3	Linear	10	3	Expo		J
68	GosxEP	3	Linear	10	3	Kehrw		J
69	GosxEP	3	Linear	10	3	Linear	10	J
70	GosxEP	3	Linear	10	3	TanH		J
71	GosxEP	3	TanH		3	Logarith		
72	GosxEP	3	TanH		3	Expo		
73	GosxEP	3	TanH		3	Kehrw		
74	GosxEP	3	TanH		3	Linear	10	
75	GosxEP	3	TanH		3	TanH		
76	EPGOSx	3	Linear	10	3	Logarith		J
77	EPGOSx	3	Linear	10	3	Expo		J
78	EPGOSx	3	Linear	10	3	Kehrw		J
79	EPGOSx	3	Linear	10	3	Linear	10	J
80	EPGOSx	3	Linear	10	3	TanH		J
81	EPGOSx	3	Logarith		3	Logarith		
82	EPGOSx	3	Logarith		3	Expo		
83	EPGOSx	3	Logarith		3	Kehrw		
84	EPGOSx	3	Logarith		3	Linear		
			IN			**OUT**		
Dateien	Abhäng.	Anzahl	Funktion	Divisor	Anzahl	Funktion	Divisor	Vol_Nor
85	EPGOSx	3	Logarith		3	TanH		

86	EPGOSx	3	Expo		3	Logarith		
87	EPGOSx	3	Expo		3	Expo		
88	EPGOSx	3	Expo		3	Kehrw		
89	EPGOSx	3	Expo		3	Linear	10	
90	EPGOSx	3	Expo		3	TanH		
91	EPGOSx	3	Kehrw		3	Logarith		
92	EPGOSx	3	Kehrw		3	Expo		
93	EPGOSx	3	Kehrw		3	Kehrw		
94	EPGOSx	3	Kehrw		3	Linear	10	
95	EPGOSx	3	Kehrw		3	TanH		
96	EPGOSx	3	TanH		3	Logarith		
97	EPGOSx	3	TanH		3	Expo		
98	EPGOSx	3	TanH		3	Kehrw		
99	EPGOSx	3	TanH		3	Linear	10	
100	EPGOSx	3	TanH		3	TanH		

Tabelle 10.12 Funktionszusammenhänge der Pattern- und Ergebnisfiles

11 Statistische Datenanalyse

Mit der statistischen Betrachtung sowohl der Eingabe- als auch Ausgabewerte ist es möglich durch den Korrelationskoeffizienten nach Pearson die Zusammenhänge verschiedener Attribute genauer zu analysieren.
Die Auswertungen befassen sich mit der einfachen, der zweifachen und der multiplen linearen Regressionsanalyse. Die Analysen werden mit einer Irrtumswahrscheinlichkeit von 0,001 und dem daraus folgenden Konfidenzintervall von [Mittelwert $\pm$ Konfidenz] durchgeführt.
Die Daten werden den statistischen Systemen zum einen reduziert und zum anderen komplett übergeben, wobei die Reduzierung die Skalen der Glasgow Outcome Scores und der evozierten Potentiale betrifft.

11.1 Einfache Lineare Regressions-/Korrelationsanalyse

Die Analyse auf einfache lineare Korrelation befasst sich mit folgenden Abhängigkeiten:

- GOSy in Abhängigkeit von GOSx
  Mit Hilfe dieser statistischen Betrachtung soll geklärt werden, inwieweit die Werte bei Entlassung, nach 3 Monaten und nach 12 Monaten untereinander korrelieren.
  Innerhalb dieser Auswertung stellt sich heraus, dass die Werte des GOS3M und GOS12M je Patient die gleichen Zahlen annehmen, was einen Pearsonkorrelationskoeffizienten von 1,0 zur Folge hat.
  Die Korrelation zwischen den Merkmalen GOSE und GOS3M bzw. GOSE und GOS12M beträgt in der reduzierten Skala 0,676 und in der kompletten Skala 0,724.

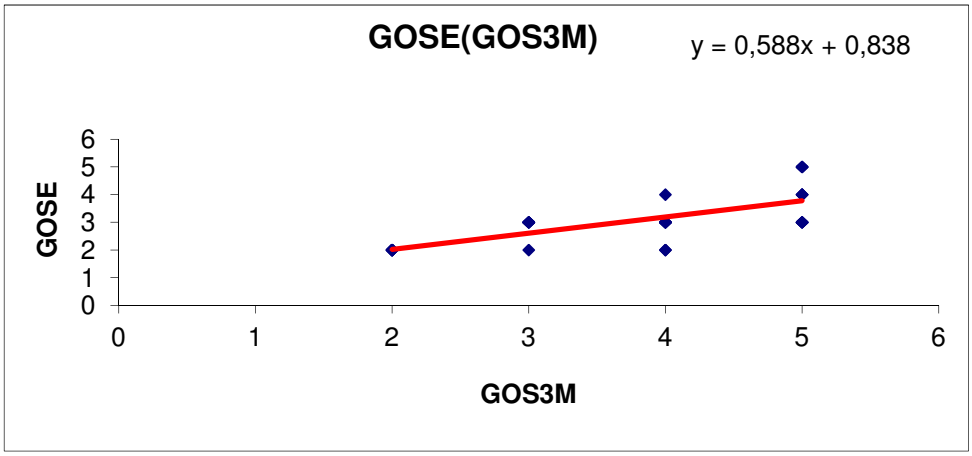


Abbildung 11.1 Graph zu GOSE = f (GOS3M) mit kompletter Skala

- yEP in Abhängigkeit von xEP

In dieser Analyse werden die Zusammenhänge der evozierten Potentiale, der Bereiche Akustik, Medianus und Tibialis, innerhalb beider Skalen betrachtet.
Als Ergebnis stellt sich heraus, dass die Korrelation zwischen MEP und TEP bei ca. 0,837 liegt (Abbildung 11.2), während die Korrelationen zwischen MEP und AEP bei 0,553 für die reduzierte Skala und 0,534 für die komplette Skala betragen.

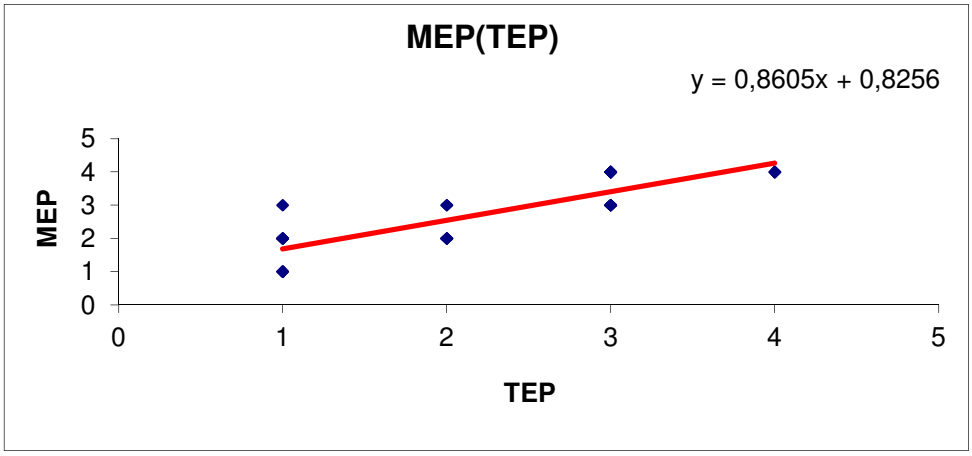


Abbildung 11.2 Graph zu MEP = f (TEP) mit kompletter Skala

Mit der reduzierten Skala ergibt sich für die Korrelation zwischen TEP und AEP ein Wert von 0,445, der durch Komplettierung der Skala auf 0,412 reduziert wird (Abbildung 11.3).

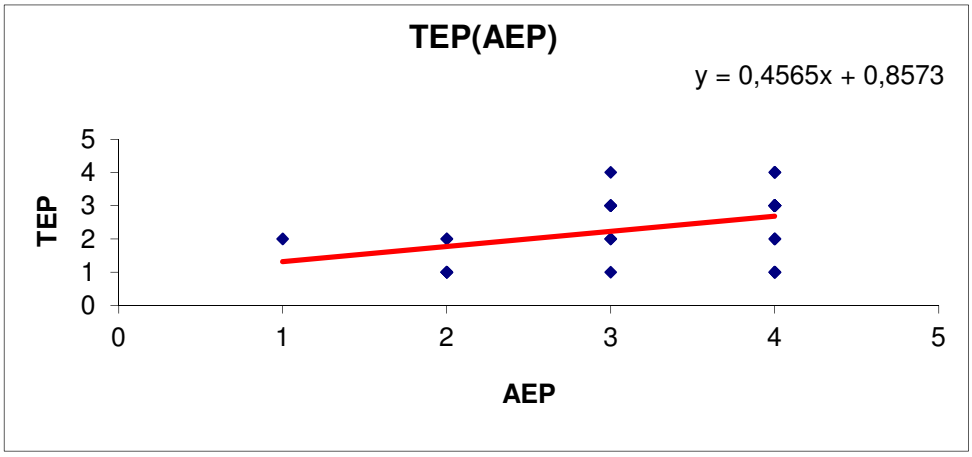


Abbildung 11.3 Graph zu TEP = f (AEP) mit kompletter Skala

- GOSy in Abhängigkeit von xEP

Diese Zuordnung beschreibt den Zusammenhang der einzelnen Glasgow Outcome Scores zwischen jedem evozierten Potential.

Die beste Regression wird im Bereich der reduzierten Skala durch die lineare Funktion GOSE = f (TEP) mit einem Korrelationskoeffizienten von 0,636 erreicht, während die schlechteste Korrelation bei 0,196 zwischen GOS3M und AEP liegt.

Für die komplette Skala besitzt die Abhängigkeit zwischen GOS3M/12M und TEP mit 0,749 (Abbildung 11.4) die höchste Korrelation.

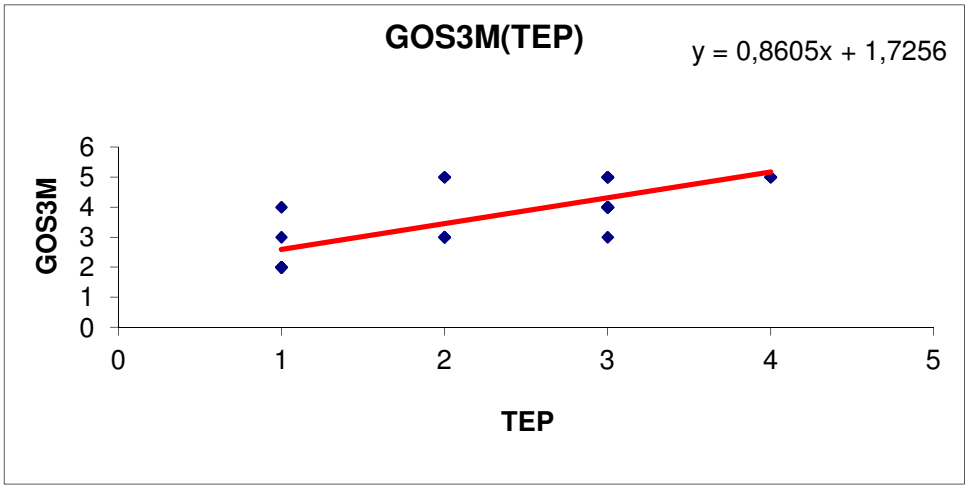


Abbildung 11.4 Graph zu GOS3M/12M = f (TEP) mit kompletter Skala

Für die Abhängigkeit von GOS3M/12M zu AEP ergibt sich der schlechteste Korrelationskoeffizient nach Pearson von 0,164 (Abbildung 11.5).

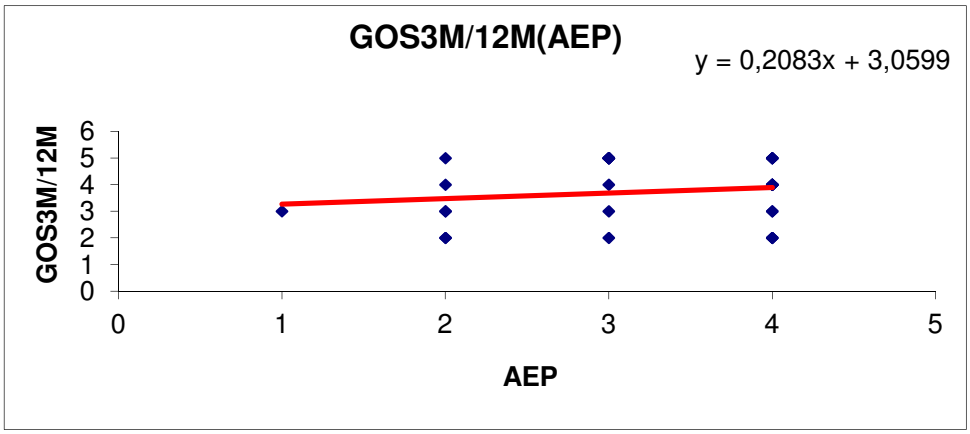


Abbildung 11.5 Graph zu GOS3M/12M = f (AEP) mit kompletter Skala

Um weitere Aussagen bzgl. der Abhängigkeiten in diesem Bereich treffen zu können, wird ein Filter auf die referentiell hinterlegten Schlüssel für Art und Lage gelegt. Die einzelnen Korrelationskoeffizienten können bzgl. der kompletten sowie reduzierten Skala den Tabellen 11.1 bis 11.4 entnommen werden.

Für die Lage einer Läsion stehen insgesamt sieben Ausprägungen zur Verfügung, für die Art einer Läsion sind es vier.

Die besten Korrelationen werden besonders hervorgehoben. Der maximale Korrelationskoeffizient liegt bei 0,926 für den Zusammenhang zwischen GOS3M/12M(TEP) in der Lage „Capsula interna“. Das Minimum der Korrelation befindet sich bei 0,000 für die Abhängigkeit von GOS3M/12M zu AEP in der Lage „Capsula interna“.

Die Anzahl der vorhandenen Datensätze, die nach Filtrierung der Analyse zur Verfügung stehen, werden in der Spalte „N“ dargestellt.

Für die Korrelation im Bereich der Art „Diffuse hypoxic damage“ kann keine Analyse durchgeführt werden, da diese Daten lediglich einen Patienten betreffen.

GOSE	N	AEP	MEP	TEP
Alle	**30**	0,239	0,589	0,735
Nach Lage:				
Mesencephalon Pons Medulla	**22**	0,084	0,544	0,711
Frontalis	**26**	0,266	0,663	0,793
Parietalis occipitalis	**15**	**0,343**	0,536	0,761
Insularis temporalis	**20**	0,342	**0,692**	**0,802**
Corpus callosum Septum	**21**	0,052	0,439	0,641
Capsula interna	**10**	0,214	0,551	0,650
Nuclei basales Limbisches System	**24**	0,235	0,609	0,729
Nach Art:				
Diffuse hypoxic damage	**1**	n.b.	n.b.	n.b.
Hematoma	**11**	**0,291**	**0,637**	**0,828**
Contusion Cerebrum	**30**	0,239	0,589	0,735
Contusion Truncus encephalicus	**26**	0,144	0,493	0,640

Tabelle 11.1 Korrelation zwischen GOSE und xEP komplette Skala

GOS3M / GOS 12 M	N	AEP	MEP	TEP

Alle	**30**	0,164	0,671	0,749
Nach Lage:				
Mesencephalon Pons Medulla	**22**	0,124	0,636	0,738
Frontalis	**26**	0,266	0,783	0,817
Parietalis occipitalis	**15**	0,302	0,725	0,866
Insularis temporalis	**20**	**0,342**	0,661	0,743
Corpus callosum Septum	**21**	0,067	0,601	0,723
Capsula interna	**10**	0,000	**0,875**	**0,869**
Nuclei basales Limbisches System	**24**	0,127	0,684	0,784
Nach Art:				
Diffuse hypoxic damage	**1**	n.b.	n.b.	n.b.
Hematoma	**11**	**0,261**	**0,804**	**0,873**
Contusion Cerebrum	**30**	0,164	0,671	0,749
Contusion Truncus encephalicus	**26**	0,102	0,613	0,711

Tabelle 11.2 Korrelations zwischen GOS3M/12M und xEP komplette Skala

GOSE	N	AEP	MEP	TEP
Alle	**30**	0,199	0,530	0,636
Nach Lage:				
Mesencephalon Pons Medulla	**22**	0,033	0,531	0,750
Frontalis	**26**	0,217	0,676	0,793
Parietalis occipitalis	**15**	**0,317**	0,532	0,758
Insularis temporalis	**20**	0,309	**0,707**	**0,816**
Corpus callosum Septum	**21**	0,052	0,439	0,641
Capsula interna	**10**	0,106	0,607	0,677
Nuclei basales Limbisches System	**24**	0,216	0,624	0,787
Nach Art:				
Diffuse hypoxic damage	**1**	n.b.	n.b.	n.b.
Hematoma	**11**	**0,269**	**0,671**	**0,828**
Contusion Cerebrum	**30**	0,199	0,530	0,636
Contusion Truncus encephalicus	**26**	0,110	0,483	0,671

Tabelle 11.3 Korrelation zwischen GOSE und xEP reduzierte Skala

GOS3M / GOS 12 M	N	AEP	MEP	TEP

Alle	**30**	0,196	0,576	0,616
Nach Lage:				
Mesencephalon Pons Medulla	**22**	0,144	0,665	0,769
Frontalis	**26**	0,252	0,741	0,798
Parietalis occipitalis	**15**	**0,397**	0,717	0,922
Insularis temporalis	**20**	0,058	0,631	0,757
Corpus callosum Septum	**21**	0,102	0,653	0,771
Capsula interna	**10**	0,000	**0,867**	**0,926**
Nuclei basales Limbisches System	**24**	0,179	0,679	0,796
Nach Art:				
Diffuse hypoxic damage	**1**	n.b.	n.b.	n.b.
Hematoma	**11**	**0,406**	**0,764**	**0,899**
Contusion Cerebrum	**30**	0,196	0,576	0,616
Contusion Truncus encephalicus	**26**	0,143	0,636	0,763

Tabelle 11.4 Korrelation zwischen GOS3M/12M und xEP reduzierte Skala

11.2 Zweifache Lineare Regressions-/Korrelationsanalyse

Innerhalb dieser Korrelationsanalyse werden die zweifachen linearen Abhängigkeiten der Glasgow Outcome Scores zu je zwei Werten der evozierten Potentiale (AEP, MEP, TEP) betrachtet.
Die Auswertungen werden sowohl für die reduzierte als auch die komplette Skala durchgeführt. Somit ergeben sich drei mögliche Funktionen, die hier mit den besten Zusammenhängen charakterisiert werden.

1. GOSy = f_1 (MEP,TEP):

   Der beste Zusammenhang wird bei der kompletten Skala erreicht. Der Korrelationskoeffizient beträgt 0,753.
   Die Funktion GOS3M/GOS12M = f_1(MEP,TEP) stellt sich dar als GOS3M/GOS12M = 0,161 · MEP + 0,722 · TEP + 1,592 dar.

2. GOSy = f_2 (AEP,TEP):

   Der höchste Korrelationskoeffizient von 0,765 wird innerhalb der kompletten Skala erreicht. Es ergibt sich für die Funktionsgleichung folgende Struktur: GOS3M/12M(AEP,TEP) = -0,222 · AEP + 0,943 · TEP + 2,251.

3. GOSy = f_3 (MEP,AEP):

   Die Gleichung GOS3M/12M= f_3 (MEP,AEP) = 0,914 · MEP – 0,346 · AEP + 2,262 stellt den besten zweifachen linearen Zusammenhang innerhalb der kompletten Skala dar, wobei der Pearsonkorrelationskoeffizient bei 0,709 liegt.

Durch eine Reduzierung der Skala ergeben sich für drei Abhängigkeiten Korrelationskoeffizienten, die um ca. 0,15 niedriger liegen als die der kompletten Skala.

11.3 Multiple lineare Regressions-/Korrelationsanalyse

Mit Hilfe dieser Auswertung wird der multiple lineare Zusammenhang in den Bereichen der Glasgow Outcome Scores, der evozierten Potentiale und der Läsionsdaten, bestehend aus den Attributen „Art“ (AID), „Lage“ (LID) und „Volumen“ (Vol), untersucht.

Es handelt sich somit um eine Funktion der Form

$$\mathbf{f(x_1,x_2,x_3) = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + d}$$

Der Zusammenhang wird in reduzierter und kompletter Form analysiert.

- yEP = f (GOSE, GOS3M, GOS12M) = a · GOSE + b · GOS3M + c · GOS12M + d

  Der Parameter „c“ nimmt innerhalb dieser Untersuchung den Wert „0“ an, da die Daten des GOS3M und GOS12M die gleichen Ausprägungen besitzen und „c“ daher bereits durch Parameter „b“ bestimmt wird.

  Es stellt sich heraus, dass der Zusammenhang des „Medianus EP“ und des „Tibialis EP“ in Bezug auf die Glasgow Outcome Scores wesentlich signifikanter ist als der des akustischen evozierten Potentials. Dieser Zusammenhang lässt sich durch die komplette Skala in allen drei Bereichen steigern. Der beste Korrelationskoeffizient von 0,799 wird für das evozierte Potential im Bereich Tibialis erreicht:

  TEP = f (GOSE, GOS3M, GOS12M) = 0,435·GOSE +0,396·GOS3M – 0,463

- GOSy = f (AEP, MEP, TEP) = a · AEP + b · MEP + c · TEP + d

  Die Abhängigkeiten der Glasgow Outcome Scores nach 3 bzw. nach 12 Monaten werden durch identische Funktionen dargestellt. Die Gründe hierfür wurden bereits beschrieben (Identität der Werte).

  Der Zusammenhang des GOSEs bzgl. der evozierten Potentiale ist in beiden Skalenbereichen nahezu identisch, d.h. reduzierte bzw. komplette Skala erzeugen den gleichen Korrelationskoeffizienten.

  Der beste Zusammenhang mit dem Korrelatioskoeffizienten von 0,617 wird innerhalb der kompletten Skala durch folgende Funktion erreicht:

  GOS3M/12M=f (AEP, MEP, TEP)=-0,581·AEP+1,394·MEP-0,123·TEP+1,948

- GOSy = f (AID, LID, VOL) = a · AID + b · LID + c · VOL + d

  Bei der Untersuchung des Zusammenhangs zwischen der GOSE–Werte und den Läsionsdaten wird die höchste lineare Korrelation von 0,067 mit Hilfe der kompletten Skala erreicht, während bei der Untersuchung der GOS3M/12M–Werte der beste Koeffizient von 0,110 mit der reduzierten Skala erreicht wird, wobei auch in diesem Fall die Funktionen identisch sind.
  Die entsprechende Funktion lautet:

  GOS3M/12M = f (AID, LID, VOL) = 0,047·AID+0,043·LID+0,002·VOL+1,836

- yEP = f (AID, LID, VOL) = a · AID + b · LID + c · VOL + d

  Durch die Reduzierung der Skala kann die Zuordnung, die durch die obige Funktion beschrieben wird, gestärkt werden. Die Korrelationswerte sind im Durchschnitt ca. 0,014 höher als innerhalb der reduzierten Skala.
  Den höchsten Pearsonkorrelationskoeffizienten von 0,157 erreicht man für das evozierte Potential im Bereich Medianus. Die Funktion lautet:

  MEP = f (AID, LID, VOL) = -0,043·AID + 0,046·LID + 0,005·VOL + 1,683

11.4 Zusammenfassung

- Einfache lineare Korrelationsanalyse:

In dieser Analyse hat sich gezeigt, dass die Abhängigkeiten der Glasgow Outcome Scores von den evozierten Potentialen des „Medianus“ und „Tibilais“ wesentlich größer sind als die Zuordnung zum akustischen evozierten Potential.
Auch die Abhängigkeiten der evozierten Potentiale untereinander liefert im akustischen Bereich den schlechtesten Koeffizienten nach Pearson.
Durch einen vorgeschalteten Filter auf die Ausprägungen der Art und Lage einer Läsion ist es möglich, den Korrelationskoeffizienten auf einen Wert von 0,926 zu erhöhen.

- Zweifache lineare Korrelationsanalyse:

Die Korrelationskoeffizienten zwischen den Glasgow Outcome Scores und zwei der drei möglichen evozierten Potentiale liegen innerhalb der kompletten Skala durchgehend bei über 0,70. Dieser Wert wird durch Reduzierung des Wertebereichs auf > 0,55 gesenkt.
Ein Problem stellen innerhalb dieser Analyse die identischen Werte nach 3 bzw. 12 Monaten des Glasgow Outconme Scores dar, da durch diese Identität die Zuweisungsmöglichkeiten auf 2 Funktionen je Abhängigkeit dezimiert werden.

- Multiple lineare Korrelationsanalyse:

Diese Auswertung erbringt in den Verkettungen der evozierten Potentiale zu den Glasgow Outcome Scores Korrelationen, die als signifikant zu beschreiben sind. Währenddessen sind innerhalb der Abhängigkeiten der Potentiale - bzw. Outcome Scores bzgl. der Läsionsdaten (Art, Lage, Volumen) - durchgehend schlechte Korrelationen zu erwarten. Die Ursache hierfür ist zum einen in der Reduzierung der Skalen für Lage und Ort und zum anderen in der Anzahl der gelieferten Daten zu suchen, die durch Einschränkung der Ausprägungen von Lage und Art von ursprünglich 473 auf 320 Datensätze gesenkt wurden.

- Interpretation der Ergebnisse:

Durch die in Kapitel 9.1 klassifizierten Gruppen und deren Problemfelder ergeben sich anhand der durchgeführten statistischen Analysen folgende Rückschlüsse:

1. *Gruppe III von Gruppe I:*

   Die Zusammenhänge zwischen den Prognosewerten (Glasgow Outcome Score) und den Läsiondaten (Art, Lage, Volumen) stellen sich durch den Korrelationskoeffizienten von unter 0,10 innerhalb der multiplen Regressionsanalyse schlecht dar.

   Diese analysierten Abhängigkeiten lassen den Rückschluss zu, dass die begrenzten Daten der Läsion bestenfalls zur Filtrierung herangezogen werden sollten bzw. die Einschränkungen in den Referenzen Art und Lage aufzuheben sind, um dadurch einen detaillierteren Wertebereich erhalten zu können.

2. *Gruppe II von Gruppe I:*

   Die Abhängigkeiten der evozierten Potentiale von den Läsionsdaten stellen sich aufgrund der multiplen Regressionsanalyse als nur schwach ausgeprägt dar.

   Diese Korrelation könnte durch einen Verzicht auf die Skalierung der evozierten Potentiale (entspricht der Verwendung von den originalen Messwerten) und einer Detaillierung der Läsionsreferenzen gesteigert werden.

3. *Gruppe III von Gruppe II:*

   Mit dieser Untersuchung wird der lineare Zusammenhang der Prognosewerte (Glasgow Outcome Score) von den Messwerten der evozierten Potentiale betrachtet.

   Es hat sich gezeigt, dass durch eine zweifache bzw. multiple Regressionsanalyse wesentlich bessere Korrelationen als eine einfache Analyse liefert.

   Als problematisch stellt sich die Identität der Progrnosewerte nach 3 bzw. 12 Monaten dar, wodurch eine Messung des Glasgow Outcome Scores nach 8 Monaten anstelle von einer Messung nach 3 und 12 Monaten zu empfehlen ist.

   Durch diese Reduzierung, könnte eine gute mathematische Abbildung mit einer zweifachen Regressionsfunktion in Abhängigkeit von den evozierten Potentialen in den Bereichen Akustik und Tibialis erreicht werden.

4. *Gruppe II von Gruppe III:*

Die multiple Regressionsanalyse ermittelt die Funktion, die die Zuordnung der evozierten Potentiale aufgrund der Glasgow Outcome Scores beschreibt.

Den besten Korrelationkoeffizienten von 0,799 innerhalb dieser beiden Bereiche erreicht man durch Abbildung des Tibialis EP mit Hilfe der Werte des Glasgow Outcome Scores.

Anhand dieses Ergebnisses lässt sich vermuten, dass die Messung des evozierten Potentials in den Bereichen Tibialis oder Medianus mit Hilfe einer Abbildungsfunktion realisiert werden könnte, wodurch sich eine primäre Erfassung erübrigen würde.

5. *Gruppe II (III) von Gruppe II (III):*

   Die einfache lineare Regressionsanalyse zeigt, dass sich die Prognosewerte des Glasgow Outcome Scores untereinander in den Bereichen nach Entlassung und 3 bzw. 12 Monaten sehr gut ableiten lassen.

   Dieses Resultat unterstreicht die Hypothese, dass es ausreichen kann, den Wert des Glasgow Outcome Scores lediglich bei Entlassung und nach 8 Monaten zu bestimmen.

   In den Bereichen der evozierten Potentiale stellt sich hearus, dass der Zusammenhang der Werte des Medianus und Tibialis stärker ausgeprägt ist als der bzgl. des akustischen evozierten Potentials.

   Dieses Ergebnis lässt den Rückschluss zu, dass der Wert des akustischen evozierten Potentials einen wichtigen Parameter darstellt, währenddessen man auf einen der beiden anderen Werte verzichten könnte, da sich der Medianus EP aus dem Tibialis EP und umgekehrt gut errechnen lässt.

Inwieweit aus den Werten der Gruppe II (evozierte Potentiale) auf die Daten der Läsion (Art, Lage und Volumen) zurückzuschließen ist, wird in dieser Analyse nicht detailiert dargestellt (siehe Problemstellung Kapitel 9.1). Es ergeben sich bei ersten Analysen der Abhängigkeiten Korrelationskoeffiezienten von < 0,2, wodurch sich im Bereich der Statistischen Regressionsanalyse eine nähere Untersuchung nicht anbietet.

12 Softwaretool KNN 1.0

12.1 Einleitung

Aus der in den zuvor beschriebenen Kapiteln hervorgehenden Problematik heraus ist das Tool KNN 1.0 entwickelt worden. Es handelt sich bei diesem Programm, das mit Hilfe von „Visual Basic 5.0“ und „Microsoft Access/Excel 97“ programmiert wurde, um ein in jedes andere Softwareprodukt integrierbares Berechnungstool.
Voraussetzung ist im Bereich der Systemumgebung „Microsoft Windows 95/98/NTx.xx“ und im Bereich der Anwendersoftware eine vollständige Installation von „Microsoft Access/Excel 97“ mit allen Datenkonvertierungsfeatures.
Die zu konvertierenden Daten müssen in einer relationalen Datenbank gehalten werden und in die Datei „KNN.MDB“ eingebunden/importiert werden, um direkt auf die zu analysierenden Daten zugreifen zu können. Weitere Informationen zur Installation befinden sich im Benutzerhandbuch zu KNN 1.0 bzw. in der Hilfedatei des Tools.

Diese Software behandelt folgende Schwerpunkte:

- *Datenkonvertierung*

  Mit Hilfe von KNN 1.0 lassen sich Daten aus einer Datenbank in das System konvertieren, sofern das vorhandene Datenbankformat unterstützt wird.
  Vor der Datenübernahme kann die Skalenart (reduziert oder komplett) gewählt werden.

- *Datentransformation*

  Diese Daten können nach erfolgreichem Import zur Erzeugung von Pattern-Files für Simulatoren künstlicher neuronaler Netze transformiert werden. KNN 1.0 erzeugt die Trainingsfiles im ASCII-Format und speichert diese benutzerdefiniert ab.

- *Datenanalyse des Inputs:*

  Damit nur relevante Korrelationen unter einem neuronalen Netz abgebildet werden können, bietet dieses Produkt die Möglichkeit die importieren Daten zuvor zu analysieren.
  Hierfür ist in KNN 1.0 ein Regressionsanalyse–Tool, mit dessen Hilfe Analysen bzgl. eines Ausgabewertes und bis zu acht Eingabewerte durchgeführt werden können, integriert.
  Als Ergebnis werden die zugehörige Funktion, das Bestimmtheitsmaß und der Korrelationskoeffizient nach Pearson errechnet.
  Handelt es sich um eine einfache, lineare Korrelationsanalyse, so wird der zugehörige Graph zusätzlich angezeigt.

- *Datenanalyse des Outputs:*

  Für die per Simulator erzeugten Ergebnis-Files bietet KNN 1.0 ein Auswertungs–Tool, mit dem die Varianzen, die Standardabweichungen und die mittleren quadratischen Fehler errechnet, gespeichert und selektiert werden können.

12.2 Benutzungsoberfläche

Nach Start des Programms öffnet sich automatisch die Benutzungsoberfläche, die in Abbildung 12.1 dargestellt ist. Mit Hilfe dieses Navigators gelangt man in alle zur Verfügung stehenden Bereiche:

- *Datenübernahme*

  Patientenübernahme importiert die Daten aus der Patientendatenbank

  Läsionenübernahme importiert die Daten aus der Läsionendatenbank

- *Patternfiles*

  Patternerzeugung öffnet Maske zur Einstellung des Datentransfers

- *Regressionsanalyse*

  Regressionsanalyse öffnet Maske zur Einstellung der Analyse

- *Ergebnisanalyse*

  SNNS-Auswertung öffnet Maske zur Analyse / Suche der Ergebnisse

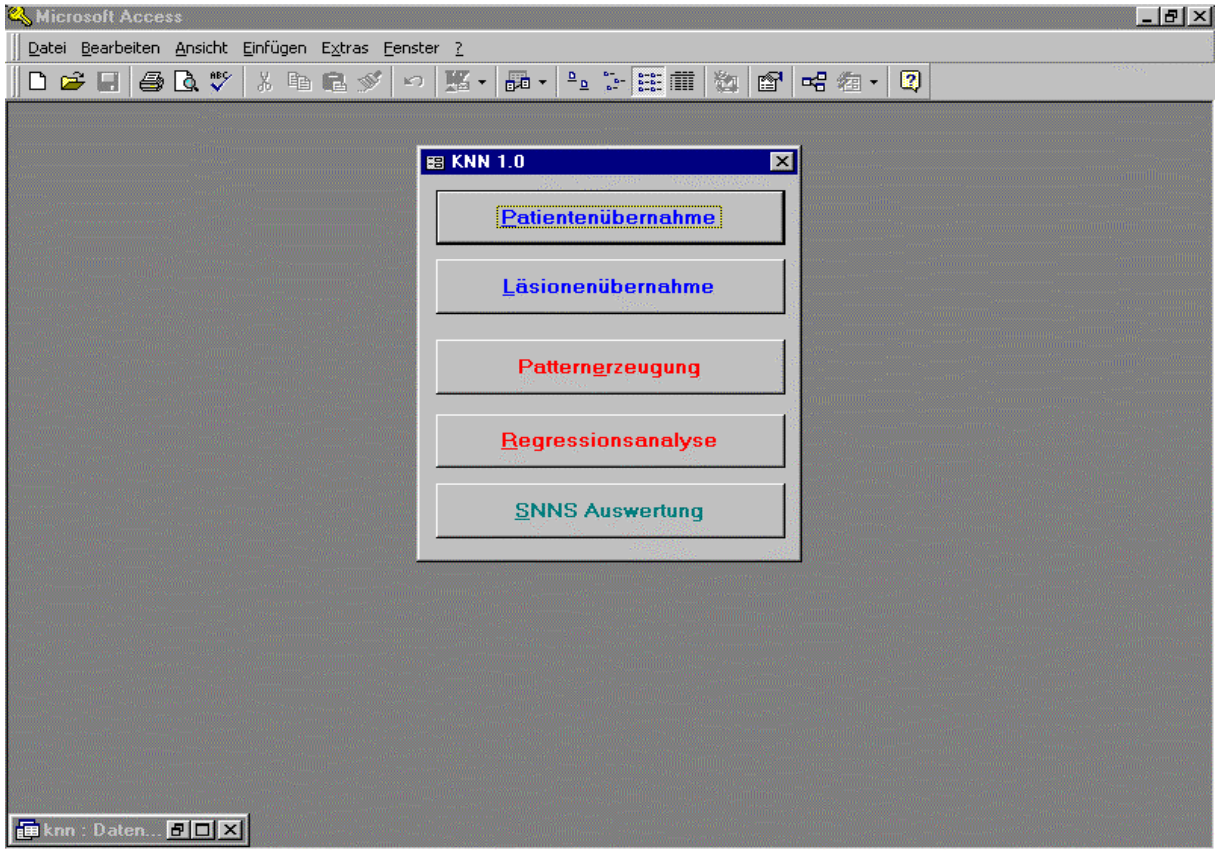


Abbildung 12.1 KNN 1.0 Benutzungsoberfläche

12.3 Datenübernahme

Mit diesen Tools werden die Daten aus der anonymisierten Datenbank der Neurochirurgie des Städtischen Klinikums Fulda in die Datenbank von KNN 1.0 konvertiert. Die Daten lassen sich in zwei Bereiche aufteilen:

- *Läsionen:*

  Innerhalb dieser Konvertierung werden die Daten aus den Tabellen der Patienten und der Läsionen entsprechend der hinterlegten Referenztabellen für „Art“ (Tabelle 9.3) und „Lage“ (Tabelle 9.2) zusammengeführt und in der KNN - Tabelle „Läsionen Neu“ gespeichert. Bei den gespeicherten Werten handelt es sich um den Patientenschlüssel (PID), Lagenummer (LID) und Artnummer (AID).

- *Patienten*

  Durch diesen Import werden die Daten aus den Patienten-, den Glasgow Outcome Score- und den Evozierten Potentiale-Bereichen zusammengeführt. Vor Start des Imports hat der Benutzer noch die Möglichkeit sich für die reduzierte oder die komplette Skalenübernahme zu entscheiden. Der Unterschied liegt im Wertebereich der Skala. Für die reduzierte Übernahme ist er auf den Bereich von [1;3] beschränkt, für die komplette Übernahme bleibt die Skala von [1;5] erhalten.
  Die Informationen werden in der KNN 1.0 Tabelle „Patienten Neu“ in den Feldern PID (Patientenschlüssel), GOSx (**x** = **E**ntlassung, nach **3M**onaten, nach **12M**onaten) und xEP (**x** = **A**kustik, **M**edialis, **T**ibialis) abgelegt.

12.4 Transformationsfunktionen

In den Bereichen der Patternerzeugung und der Regressionsanalyse können die Eingabe- und die Ausgabewerte mit Hilfe der hinterlegten Funktionen (Tabelle 12.1) transformiert werden. Durch diese eingebaute Funktion ist es möglich, den Eingabe-/Ausgaberaum einer Variablen so zu strecken bzw. zu stauchen, dass die ursprünglich schlechte Korrelation wesentlich verbessert werden kann. Wird eine solche positiv korrelierende Funktion errechnet, so können die eingesetzten Transformationsfunktionen den Datenbereich des Trainings-Files für die Simulatoren auf die gleiche Art und Weise transformieren.

Handelt es sich um eine Funktion mit integrierter Konstante, wird der vom Benutzer einzugebende Parameter, der alle Werte > 0 annehmen kann, zur Berechnung herangezogen. In der Grundeinstellung nimmt die Konstante stets den Wert 1 an. Durch diese Einstellung wird vermieden, dass Werte außerhalb des Definitionsbereichs der Funktion übergeben werden.

Die Funktionsreferenzen befinden sich in der Tabelle „Funktion“ in der Datenbank. Die Berechnung der zu transformierenden Daten wird mit Hilfe temporärer Tabellen und hinterlegter Funktionen zum Schlüssel „FID“ der selektierten Transformation realisiert.

FID	Funktion	f(x)
1	Logarithmus	ln (x+konst)
2	Exponentiell	Exp (-x)
3	Kehrwert	1 / (x+konst)
4	Linear	x / Dividend
5	TangensH	tanh (x)
6	Sinus	sin(x)
7	Wurzel	1 / Wurzel(x+konst)

Tabelle 12.1 KNN 1.0 Transformationsfunktionen:

12.5 Patternfiles

- **Allgemein:**

  Bei den Patternfiles handelt es sich um Trainingsdaten, die ein Simulator künstlicher neuronaler Netze zum Lernen und Trainieren benötigt. Die Daten werden den Neuronen auf der Eingabe- und Ausgabeschicht zur Einstellung der Gewichte der Verbindungen mit Hilfe der im Text-Format gespeicherten Datei zur Verfügung gestellt.

  Die Struktur solcher Dateien wird durch die Anordnung und Formatierung der einzelnen Daten bestimmt.

  Struktur eines Patternfiles von SNNSv4.1:

  SNNS pattern definition file V3.2
  generated at Samstag, 19. Dezember 1998

  No. of patterns : *<Anzahl aller Datensätze>*
  No. of input units : *<Anzahl der Eingabevariablen>*
  No. of output units : *<Anzahl der Ausgabevariablen>*

  Format(x_1, ##.#####) Format(x_2, ##.#####) Format(x_3, ##.#####)
  Format(y_1, ##.#####) Format(y_2, ##.#####) Format(y_3, ##.#####)

- **KNN 1.0:**

  Mit Hilfe des Moduls „Patternerzeugung“, das in Abbildung 12.2 dargestellt wird, ist es möglich über die importierten Daten die für die Simulatoren „SNNSv4.1“, „HavbpNet++“ und ausgewählten „Turbo Pascal Files“ notwendigen Trainings-Files im korrekten Format zu erstellen.

  Die hinterlegten drei unterschiedlichen Strukturen der Pattern-Files sind innerhalb der Marktforschung nach nicht kommerziellen Simulatoren am häufigsten anzutreffen.

  Die Datei wird in einem benutzerdefinierten Verzeichnis mit einem selbst definierten Dateinamen gespeichert und steht den Simulatoren zum Trainieren/Lernen zur Verfügung.

- **Einstellungen**:

Innerhalb der Maske aus Abbildung 12.2 lassen sich folgende Einstellungen unterscheiden:

- *Anzahl:*

Dieses Feld beschreibt die Anzahl der integrierten Eingabe- und Ausgabewerte. Diese Funktion ist in dieser Version noch nicht variabel gestaltet. Es können nur 3x3 Dateien erzeugt werden.

- *Volumen-Normalisierung:*

Mit Hilfe dieses Flags wird der Eingaberaum im Bereich des Volumens anhand des maximalen Wertes normalisiert. Der Wertebereich des Volumens wird dadurch auf [0;1] reduziert.

- *Simulator / Abhängigkeiten*:

Mit Hilfe dieser Schaltflächen werden der entsprechende Simulator und die gewünschte Abhängigkeit selektiert.

- *Ziel*:

Mit diesem Feld wählt der Benutzer Laufwerk, Verzeichnis und Dateinamen des zu erzeugenden Patternfiles .

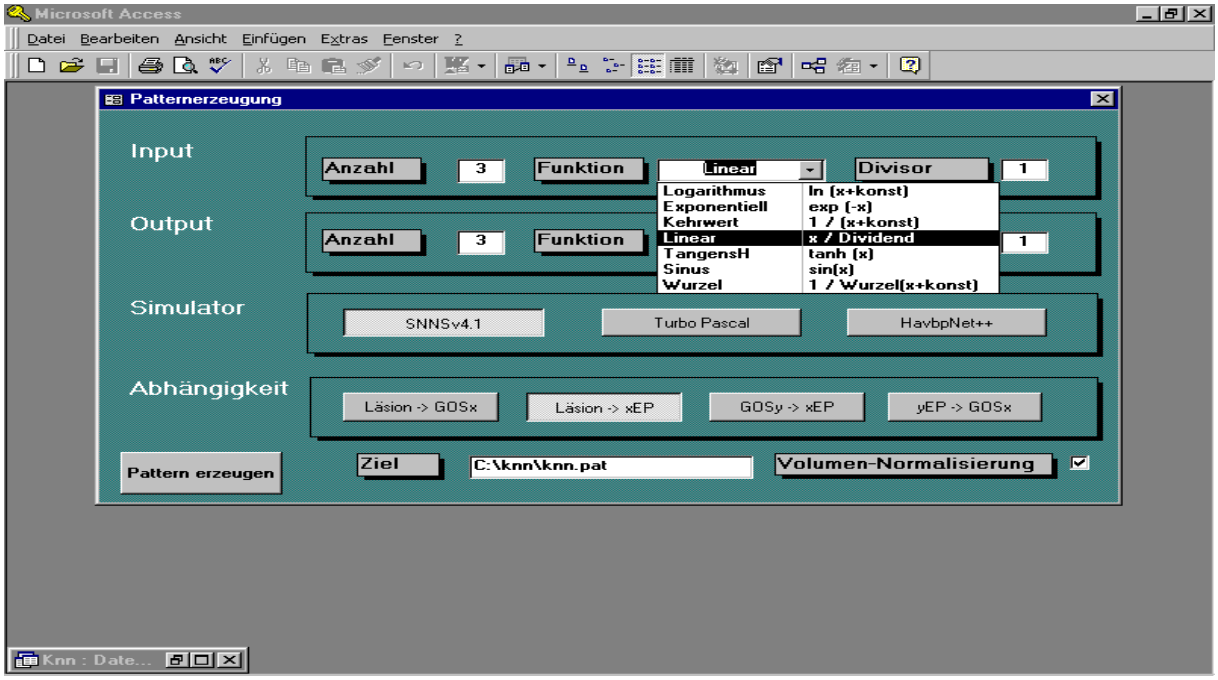


Abbildung 12.2 KNN 1.0 Patternerzeugung

12.6 Regressionsanalyse

- **Allgemein**:

  Mit Hilfe des Tools „Regressionsanalyse“ (Abbildung 12.3) können Regressionsanalysen bzgl. eines Ausgabewertes in Abhängigkeit von bis zu acht Variablen durchgeführt werden. Es besteht die Möglichkeit, den Eingabe- und den Ausgaberaum mit Hilfe von hinterlegten Funktionen zu transformieren.
  Als Ergebnis werden das Bestimmtheitsmaß, der Korrelationskoeffiezient nach Pearson, die Regressionsfunktion (incl. Transformationsfunktionen) und bei linearen Korrelationen die Funktion als Graph dargestellt.
  Wird der Parameter „Alpha“ für die Irrtumswahrscheinlichkeit im Wertebereich]0;1[angegeben, wird das Vertrauensintervall berechnet und angezeigt.

- **Einstellungen:**

  Innerhalb der in Abbildung 12.3 dargestellten Maske können folgende Einstellungen getroffen werden:

- *Klassifikation*:
  In dieser Referenzliste stehen die Attribute „in“, „out“ und „nicht relevant“ zur Verfügung. Die einzelnen Felder können entsprechend dieser Eintragungen klassifiziert werden. Dabei ist darauf zu achten, dass nur ein Feld als Output-Wert deklariert werden kann.

- *Ergebnisse*:
  Die errechneten Ergebnisse werden nach erfolgreicher Analyse im unteren Bereich des Formulars dargestellt, wobei durch Verwendung einer linearen Analyse der Funktionsgraph zusätzlich angezeigt wird.
  Die Werte werden mit drei Nachkommastellen dargestellt.

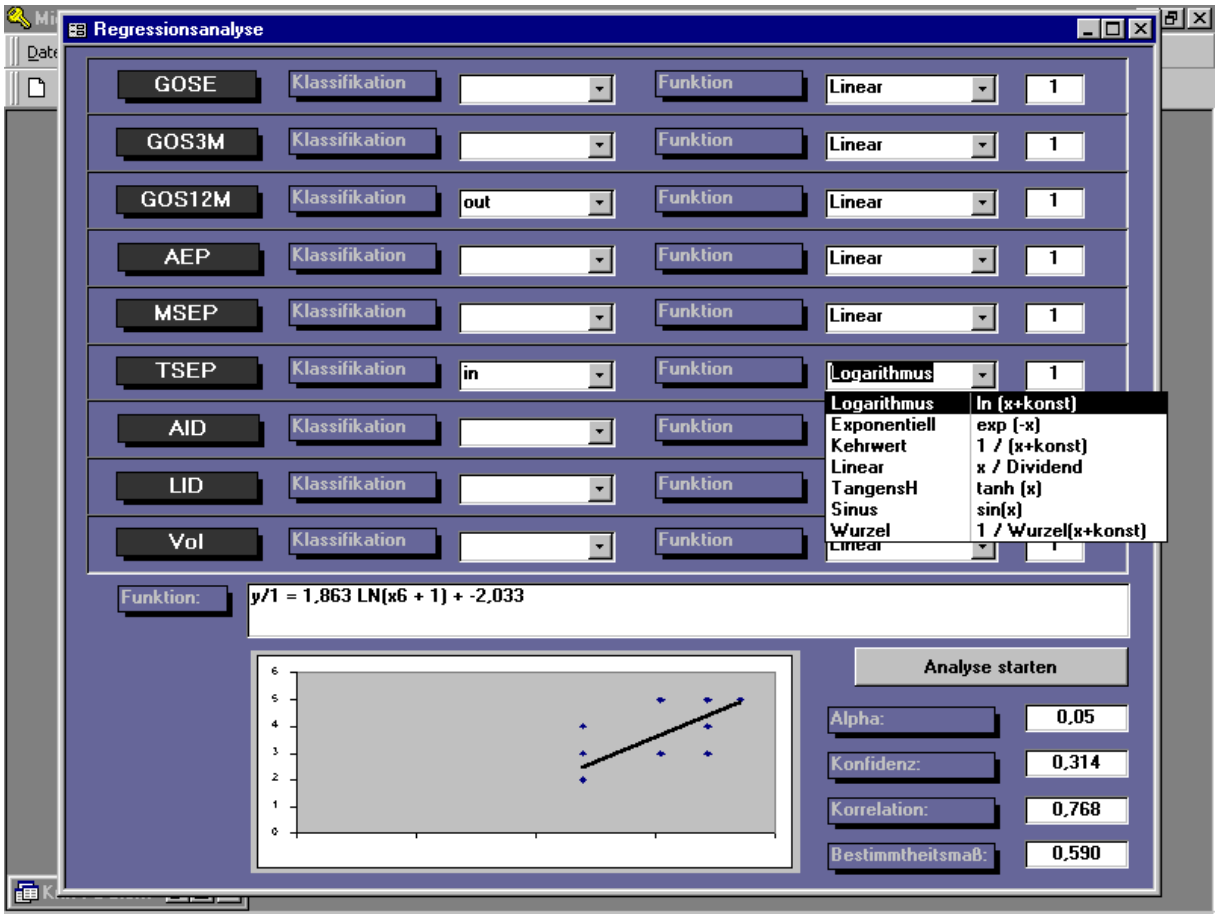


Abbildung 12.3 KNN 1.0 Regressionsanalyse

12.7 Ergebnisanalyse

Durch das in Abbildung 12.4 dargestellt Tool ist es möglich die von dem Simulator „SNNSv4.1“ errechneten Ergebnisdateien zu analysieren, zu speichern und auf diese gespeicherten Datensätze zuzugreifen. Das Tool Ergebnisanalyse errechnet die drei Varianzen, die drei Standardabweichungen und die drei mittleren quadratischen Fehler bzgl. der errechneten Ausgabewerte und der geforderten Ausgabewerte.
Voraussetzung ist eine im Textformat gespeicherte SNNS-Datei, die ohne die Titelzeilen des Files dem System übergeben werden muss.

Analyse / Suche:

Nach Auswahl des Menüpunktes „SNNS Auswertung“ wird dem Benutzer die Frage nach Durchführung der Analyse gestellt. Wird diese Frage positiv beantwortet, so analysiert das System nach Eingabe der Ergebnisdaten des Simulators die gespeicherten Werte. Durch eine negative Antwort gelangt man in das Modul, das durch Abbildung 12.5 dargestellt wird, und dort kann man gespeicherte Analysen abrufen.

- *Analyse / Speicherung:*

  Die Daten werden in der Maske (Abbildung 12.4) dargestellt und können mit Hilfe der Speichern-Schaltfläche in der KNN 1.0 Tabelle „Auswertung“ abgelegt werden. Während der Speicherung wird der Benutzer aufgefordert einen Titel für die Ergebnisanalyse zu vergeben und gleichzeitig wird der Datensatz mit einem dem aktuellen Datum als Zeitstempel versehen.

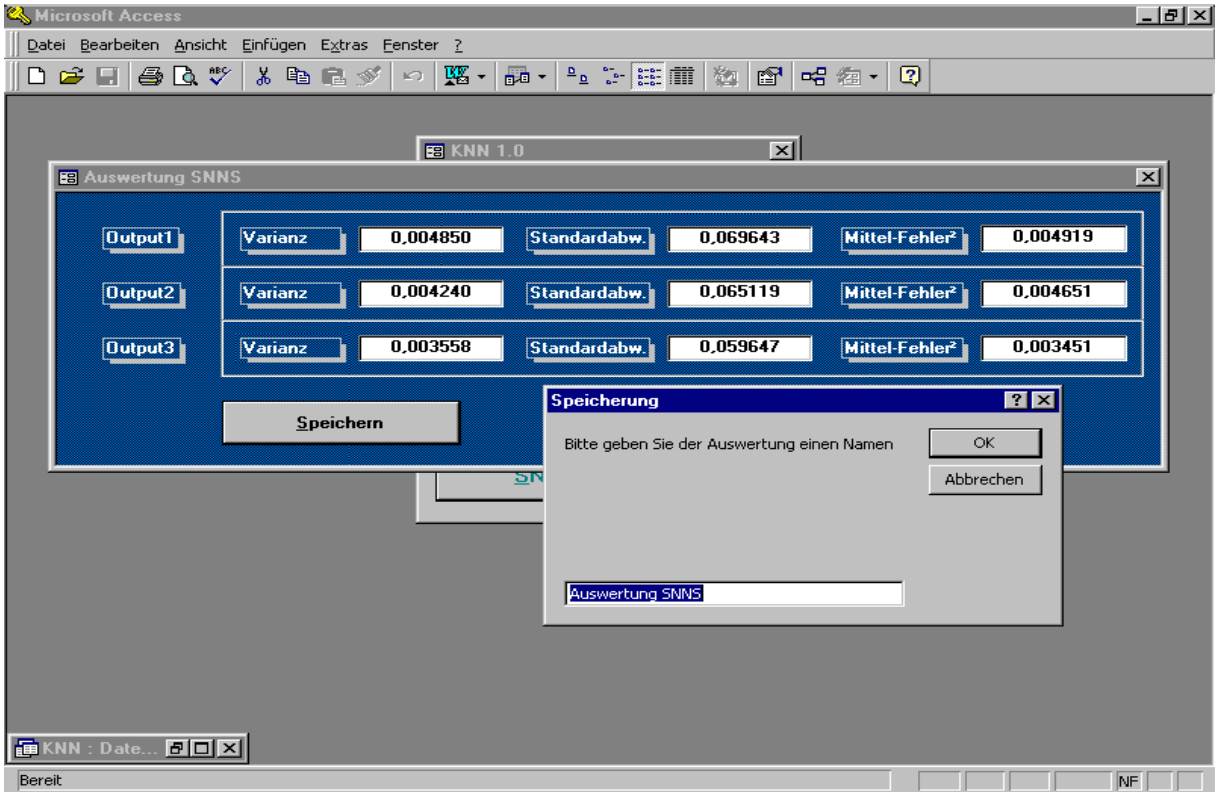


Abbildung 12.4 KNN 1.0 Auswertung SNNS

- *Analyse suchen*:

  Abbildung 12.5 stellt die Suchfunktion nach gespeicherten Ergebnisanalysen dar. Auf die hinterlegten Daten kann mit Hilfe der Suchen-Listbox zugegriffen werden. Die Selektionsfelder sind der gespeicherte Titel und der automatisch vergebene Zeitstempel bei Speicherung.

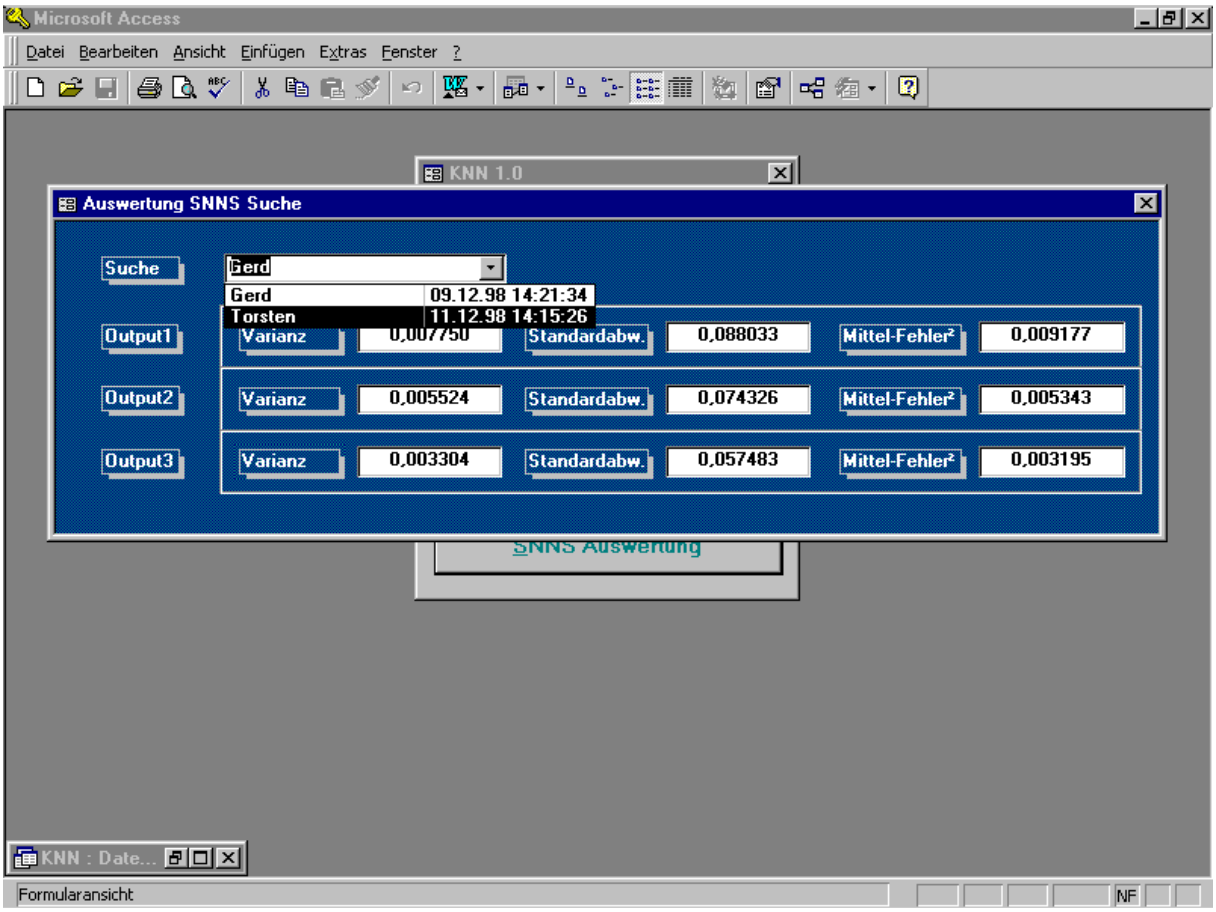


Abbildung 12.5 KNN 1.0 SNNS Auswertung suchen

13 Fazit

Die sich durch diese Arbeit ergebenden Schlussfolgerungen sind unter Berücksichtigung der geringen Anzahl an Daten zu betrachten.
Diese Datenreduzierung stellt neben dem schlechten Wertebereich die Hauptproblematik innerhalb der Forschung dar, so dass in der zukünftigen Arbeit am System des multimodularen Neuromonitorings primär auf die zur Verfügung gestellten Daten zu achten ist.

Innerhalb der Recherche nach geeigneten nicht kommerziellen Simulatoren hat sich herausgestellt, dass diese Produkte jeweils nur begrenzte Berechnungsmöglichkeiten im Hinblick auf Topologien oder die integrierten Funktionen bieten, so dass sich eine Recherche im kommerziellen Bereich anschließen sollte.

Der Stuttgarter Simulator „SNNS“ in der Version 4.1 für Windows stellt nahezu alle Topologien, Einstellungsparameter, Lernregeln und Funktionen zum Simulieren neuronaler Netze zur Verfügung. Ein Nachteil ist in der aktuellen Version der auf [0;1] begrenzte Ausgaberaum.
Dadurch ist eine Transformation der Eingabewerte notwendig geworden., damit die zu verarbeitenden Daten in dem definierten Wertebereich des SNNS v.41 liegen.
Um diese Transformationen vornehmen zu können, wurde ein Tool (KNN 1.0) entwickelt, welches die Daten in den jeweils gewünschten Datenraum überführt.
Dies stellt aber nur einen Teil des Funktionsumfangs von KNN 1.0 dar. Das Programm übernimmt die vom Klinikum gestellten Daten und arbeitet die Tabellen in eine der Normalformlehre entsprechenden Datenbank ein. Dabei kann der Benutzer wählen, ob die Daten in der Ursprungsskala oder in einer reduzierten Skala übernommen werden.
Anschließend kann der Benutzer eine Regressionsanalyse der zur Verfügung stehenden Daten durchführen, wobei die zu untersuchenden Parameter zusätzlich durch die schon erwähnten Funktionen transformiert werden können.
Ein weiteres Feature des Tools ist es, für die beschriebenden Software-Simulatoren künstlicher neuronaler Netze die entsprechenden Patternfiles zu erstellen, wobei der Ein- und Ausgaberaum normalisiert und transformiert dargestellt werden kann.

Als abrundende Funktionalität wurde ein Analysetool implementiert, welches die Ergebnisse des SNNS v4.1 nach den Soll- und Ist-Werten interpretiert und das Abspeichern der quadratischen Fehler, der Standardabweichungen und der Varianzen zulässt.
Somit ist der Zweck des Programms KNN 1.0 wie folgt zu beschreiben:

- Datenübernahme der Klinikdaten
- Bereitstellung der Simulatorendaten
- Regressionsanalyse der Klinikdaten
- Auswertung der Simulatorergebnisse

Für die Weiterentwicklung des Programms bietet sich die Dynamisierung des Eingabe- und des Ausgaberaums innerhalb der Datenübernahme an. Auch die Integration des Tools in bestehende Datenverarbeitungssoftware stellt ein Primärziel für zukünftige Arbeiten dar.
Die Rücktransformation der von SNNS v4.1 gelieferten Ergebnisdaten in den Wertebereich des Klinikums stellt eine weitere zukünftige Programmerweiterung dar.

In den Analysen mittels SNNS v4.1 ergaben sich zusammenfassend folgende Resultate:

- Die Daten der Gruppe I (Läsionsdaten) prognostizieren die Werte der Gruppe III (Glasgow Outcome Score) am besten durch die Struktur eines partiell rekurrenten künstlichen neuronalen Netzes (hier: Elman-Netz)
- Der Zusammenhang zwichen den Daten der Gruppe I (Läsionsdaten) und der Gruppe II (evozierte Potentiale) wird durch die Verwendung eines feedforward Netzes am besten abgebildet.
- Um von den Werten der Gruppe II (evozierte Potentiale) auf die Prognosewerte der Gruppe III (Glasgow Outcome Score) schließen zu können, zeigte sich ebenfalls, dass die Struktur eines feedforward Netzes zu verwenden ist.
- Rückwirkende Schlüsse der Gruppe III (Glasgow Outcome Score) auf die Gruppe II (evozierten Potentiale) können am besten durch eine feedfordward Struktur (3-32-16-8-5-3) erreicht werden.

Die verwendeten Transformationsfunktionen und die erzeugten Ergebnisse (Varianzen) können aus Kapitel 10.5 entnommen werden.

Anhand dieser Resultate stellen (bei Verwendung von korrekten Daten) die Topologien der partiell rekurrenten Netze bzw. der vollständig rekurrenten Netze vermutlich eine geeignete Struktur dar.
Die weiteren Forschungen im Hinblick auf die Verwendung nicht kommerzieller Simulatoren sollte sich mit den Bereichen der selbstorganisierenden Karten zur Musterklassifizierung und den Jordan-Elman-Netzen zur Mustererkennung befassen.

Die statistische Analyse der Abhängigkeiten von den Daten der Patienten und deren Läsionen ergab bei den Glasgow Outcome Scores und den evozierten Potentialen gute, jedoch im Bereich der Läsionendaten schlechte Korrelationskoeffizienten.
Diese Verschlechterung des Zusammenhangs liegt zum einen an der Reduzierung des Wertebereichs der Glasgow Outcome Scores und zum anderen an der Struktur der hinterlegten Skala.
Durch eine Detaillierung der Formate des evozierten Potentials auf die originalen Meßwerte und eine Skala von [1;5] für die Glasgow Outcome Scores ist es denkbar, dass sich der Zusammenhang der Daten steigern lässt.
Eine Filtrierung der Daten anhand der Referenzen bestehend aus Lage bzw. Art wirkt sich positiv auf die Korrelationen aus, so dass eine Klassenbildung der Form „Lage-Art“ die Prognosemöglichkeiten steigern könnte.
Aufgrund der inneren Abhängigkeiten der evozierten Potentiale und der Glasgow Outcome Scores, sollten die Prognosewerte nach 3 bzw. 12 Monaten durch einen Wert z.B. nach 8 Monaten ersetzt werden. Gleichzeitig könnte durch eine geeignete Abbildungsfunktion einer der beiden Messwerte der evozierten Potentiale (Tibialis und Medianus) durch den jeweilig anderen berechnet werden und somit die Erfassung auf zwei Werte (AEP und TEP oder MEP) beschränkt werden.

Die Phase des Trainierens und des Testens mit Hilfe der Softwaresimulatoren hat gezeigt, dass durch Transformation des Eingabe-/Ausgaberaums zwar keine optimalen, jedoch bessere Ergebnisse erzielt werden können.
Dies hat zur Folge, dass vor dem Training des Netzes geeignete Funktionen gewählt werden müssen, was mit Hilfe des Tools „KNN 1.0“ menügesteuert möglich ist.

Abschließende Worte :

Auch wenn es im Sinne der Forschung wichtig ist, die Prognosen über Krankheitsverlauf möglichst exakt abschätzen zu können, so bleibt doch zu bedenken, dass zielsichere Prognosen auch Probleme nach sich ziehen werden.
Dies sind z.B. der ethische Aspekt und finanzielle Einsparungen zu Lasten des Patienten. Andererseits ließen sich in Zukunft durch geeignete Abbildungen eventuell für den Patienten anstrengende Untersuchungen vermeiden, da man die Werte prognostizieren könnte.
Im Hinblick auf den rasanten technischen Fortschritt, sollten sich alle Experten einig sein, diese technische Weiterentwicklung immer zum Wohle des Menschen einzusetzen.

Gerhard Röhrig & Torsten Schreiber

14 Quellcode zu KNN 1.0

14.1 Modul Allgemein

```
Option Compare Database
Option Explicit

Public Function strReplace(ByVal strorg As String, strsearch As String, strtoreplace As String) As
String
' Mit dieser Function ist es möglich, ein beliebiges Zeichen bzw. String
' durch einen anderes(n) zu ersetzen
' storg          = Original Zeichenkette
' strsearch      = Zu suchendes Zeichen
' strtoreplace   = einzusetzendes Zeichen

Dim n As Integer

For n = 1 To Len(strorg)
   If Mid(strorg, n, 1) = strsearch Then
      strReplace = strReplace & strtoreplace
      Else
      strReplace = strReplace & Mid(strorg, n, 1)
   End If
Next n
End Function

Public Function Pat_gos_Havbpett(i As String, o As String, file As String, i_faktor As String,
o_faktor As String, In_fkt As String, Out_fkt As String)
   'Function erzeugt Pattern-File für Havbpett Konstante: Datei, input/output units
   'Formal: 3xInput 3xOutput

   'i           = Anzahl der Input Variablen
   'o           = Anzahl der Output Variablen
   'file        = Zielverzeichnis & Zieldatei
   'i_faktor    = Konstante zur Funktionsbestimmung im Eingaberaum
   'o_faktor    = Konstante zur Funktionsbestimmung im Ausgaberaum
   'In_fkt      = Schlüssel der Input-Funktion
   'Out_fkt     = Schlüssel der Output-Funktion

   Dim db As Database
   Dim nn As Recordset
   Dim dat As Recordset
   Dim pat As Recordset
   Dim sql, datei, max
   Set db = CurrentDb()
   Set nn = db.OpenRecordset("Läsionen Neu")

   datei = file
   Dim fn As Integer
   fn = FreeFile
   Open datei For Output As #fn                    'Öffnen der Ausgabedatei

   Print #fn, DCount("aid", "Läsionen Neu") / 2 & " " & i & " " & o


If Forms![Pattern]![Normal] = True Then            'Wenn zu normalisieren ist
      max = Normalize()                            'Normalisierung der Volumenwerte
```

```
    Else
    max = "1"
End If

nn.MoveFirst
Do Until nn.EOF
    Select Case In_fkt                                  'Selektion auf die Inputfunkiton
    Case "linear"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Trim(Str(nn!aid) / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Trim(Str(nn!lid) / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn![Läsion Volumen] / max), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!gose / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!gos3m / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!gos12m / o_faktor), "###0.00000"), ",", ".")
    Case "Kehrwert"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / (nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!lid + 1), "###0.00000"), ",", "."); " "; strReplace(Format((1 /
        ((nn![Läsion Volumen] / max) + 1)), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        (pat!gose + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / (pat!gos3m +
        o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / (pat!gos12m + o_faktor),
        "###0.00000"), ",", ".")
    Case "Logarithmus"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Log(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!lid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!gos12m + o_faktor), "###0.00000"), ",", ".")
    Case "Exponentiell"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Exp(-nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn![Läsion Volumen] / max)), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-pat!gose),
        "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-pat!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-pat!gos12m), "###0.00000"), ",", ".")
    Case "TangensH"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(TanH(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!lid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn![Läsion Volumen] / max), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!gos12m), "###0.00000"), ",", ".")
    Case "Sinus"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Sin(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn![Läsion
        Volumen] / max)), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(pat!gose),
        "###0.00000"), ",", "."); " "; strReplace(Format(Sin(pat!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(pat!gos12m), "###0.00000"), ",", ".")
    Case "Wurzel"
```

```
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / Sqr(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        Sqr((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        Sqr(pat!gose + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(pat!gos3m +
        o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(pat!gos12m + o_faktor),
        "###0.00000"), ",", ".")
    End Select
    nn.MoveNext
  Loop
  pat.Close
  nn.Close
  Close fn
```

End Function

Public Function Pat_gos_Pascal(i As String, o As String, file As String, i_faktor As String, o_faktor As String, In_fkt As String, Out_fkt As String)

```
  'Function erzeugt Pattern-File für Turbo Pascal Konstante: Datei, input/output units
  'Formal: 3xInput 3xOutput

  'i          = Anzahl der Input Variablen
  'o          = Anzahl der Output Variablen
  'file       = Zielverzeichnis & Zieldatei
  'i_faktor   = Konstante zur Funktionsbestimmung im Eingaberaum
  'o_faktor   = Konstante zur Funktionsbestimmung im Ausgaberaum
  'In_fkt     = Schlüssel der Input-Funktion
  'Out_fkt    = Schlüssel der Output-Funktion


  Dim db As Database
  Dim nn As Recordset
  Dim dat As Recordset
  Dim pat As Recordset
  Dim sql, datei, max
  Set db = CurrentDb()
  Set nn = db.OpenRecordset("Läsionen Neu")

  datei = file
  Dim fn As Integer
  fn = FreeFile
  Open datei For Output As #fn                          'Öffnen der Ausgabedatei
                                                        'Erstellung der Kopfzeile für Pacal
  Print #fn, Trim(DCount("aid", "Läsionen Neu") / 2)    'Anzahl der Datensätze
  Print #fn, i
  Print #fn, Trim(DCount("aid", "Läsionen Neu") / 2)
  Print #fn, o

  If Forms![Pattern]![Normal] = True Then               'Volumennormalisierung
    max = Normalize()
    Else
    max = "1"
  End If

  nn.MoveFirst
  Do Until nn.EOF
    Select Case In_fkt                                  'Selektion der Eingabefunkion
    Case "linear"
```

```
        Print #fn, strReplace(Format(Trim(Str(nn!aid) / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Trim(Str(nn!lid) / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format((pat!gose / o_faktor), "###0.00000"), ",", "."); " ";
        StrReplace(Format((pat!gos3m / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!gos12m / o_faktor), "###0.00000"), ",", ".")
    Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!lid + 1), "###0.00000"), ",", "."); " "; strReplace(Format((1 /
        ((nn![Läsion Volumen] / max) + 1)), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / (pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (pat!gos3m + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1
        / (pat!gos12m + o_faktor), "###0.00000"), ",", ".")
    Case "Logarithmus"
        Print #fn, strReplace(Format(Log(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!lid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Log(pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!gos12m + o_faktor), "###0.00000"), ",", ".")
    Case "Exponentiell"
        Print #fn, strReplace(Format(Exp(-nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn![Läsion Volumen] / max)), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Exp(-pat!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-pat!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        pat!gos12m), "###0.00000"), ",", ".")
    Case "TangensH"
        Print #fn, strReplace(Format(TanH(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!lid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(TanH(pat!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!gos12m), "###0.00000"), ",", ".")
    Case "Sinus"
        Print #fn, strReplace(Format(Sin(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn![Läsion
        Volumen] / max)), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Sin(pat!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(pat!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(pat!gos12m), "###0.00000"), ",", ".")

    Case "Wurzel"
        Print #fn, strReplace(Format(1 / Sqr(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        Sqr((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
```

```
        Print #fn, strReplace(Format(1 / Sqr(pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(pat!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(pat!gos12m + o_faktor), "###0.00000"), ",", ".")
    End Select
    nn.MoveNext
Loop
pat.Close
nn.Close
Close fn
```

End Function

Public Function Pat_ep_Pascal(i As String, o As String, file As String, i_faktor As String, o_faktor As String, In_fkt As String, Out_fkt As String)

```
'Function erzeugt Pattern-File für Turbo Pascal Konstante: Datei, input/output units
'Formal: 3xInput 3xOutput

'i          = Anzahl der Input Variablen
'o          = Anzahl der Output Variablen
'file       = Zielverzeichnis & Zieldatei
'i_faktor   = Konstante zur Funktionsbestimmung im Eingaberaum
'o_faktor   = Konstante zur Funktionsbestimmung im Ausgaberaum
'In_fkt     = Schlüssel der Input-Funktion
'Out_fkt    = Schlüssel der Output-Funktion

Dim db As Database
Dim nn As Recordset
Dim dat As Recordset
Dim pat As Recordset
Dim sql, datei, max
Set db = CurrentDb()
Set nn = db.OpenRecordset("Läsionen Neu")

datei = file
Dim fn As Integer
fn = FreeFile
Open datei For Output As #fn                        'Öffnen der Ausgabedatei
                                                    'Kopfzeile der Pascalfiles
Print #fn, Trim(DCount("aid", "Läsionen Neu") / 2)   'Anzahl der Eingabedatensätze
Print #fn, i
Print #fn, Trim(DCount("aid", "Läsionen Neu") / 2)
Print #fn, o

If Forms![Pattern]![Normal] = True Then             'Volumennormalisierung
    max = Normalize()
    Else
    max = "1"
End If

nn.MoveFirst
Do Until nn.EOF
Select Case In_fkt                                  'Selektion der Eingabefunktion
    Case "linear"
        Print #fn, strReplace(Format((nn!aid / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!lid / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
```

```
        Print #fn, strReplace(Format((pat!aep / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!msep / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!tsep / o_faktor), "###0.00000"), ",", ".")
    Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((1 /
        ((nn![Läsion Volumen] / max) + i_faktor)), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / (pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (pat!msep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        (pat!tsep + o_faktor), "###0.00000"), ",", ".")
    Case "Logarithmus"
        Print #fn, strReplace(Format(Log(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!lid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Log(pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!msep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!tsep + o_faktor), "###0.00000"), ",", ".")
    Case "Exponentiell"
        Print #fn, strReplace(Format(Exp(-nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn![Läsion Volumen] / max)), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Exp(-pat!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-pat!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        pat!tsep), "###0.00000"), ",", ".")
    Case "TangensH"
        Print #fn, strReplace(Format(TanH(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!lid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(TanH(pat!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!tsep), "###0.00000"), ",", ".")
    Case "Sinus"
        Print #fn, strReplace(Format(Sin(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn![Läsion
        Volumen] / max)), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Sin(pat!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(pat!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(pat!tsep),
        "###0.00000"), ",", ".")



    Case "Wurzel"
        Print #fn, strReplace(Format(1 / Sqr(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        Sqr((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / Sqr(pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(pat!msep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(pat!tsep + o_faktor), "###0.00000"), ",", ".")
```

```
    End Select
    nn.MoveNext
  Loop
  pat.Close
  nn.Close
  Close fn

End Function

Public Function Pat_ep_havbpett(i As String, o As String, file As String, i_faktor As String,
o_faktor As String, In_fkt As String, Out_fkt As String)
  'Function erzeugt Pattern-File für Havbpett Konstante: Datei, input/output units
  'Formal: 3xInput 3xOutput

  'i          = Anzahl der Input Variablen
  'o          = Anzahl der Output Variablen
  'file       = Zielverzeichnis & Zieldatei
  'i_faktor   = Konstante zur Funktionsbestimmung im Eingaberaum
  'o_faktor   = Konstante zur Funktionsbestimmung im Ausgaberaum
  'In_fkt     = Schlüssel der Input-Funktion
  'Out_fkt    = Schlüssel der Output-Funktion

  Dim db As Database
  Dim nn As Recordset
  Dim dat As Recordset
  Dim pat As Recordset
  Dim sql, datei, max
  Set db = CurrentDb()
  Set nn = db.OpenRecordset("Läsionen Neu")

  datei = file
  Dim fn As Integer
  fn = FreeFile
  Open datei For Output As #fn                    'Öffnen der Ausgabedatei

                                                  'Erzeugung der Kopfzeile für HavbpNet
  Print #fn, DCount("aid", "Läsionen Neu") / 2 & " " & i & " " & o

  If Forms![Pattern]![Normal] = True Then         'Volumennormaliiserung
    max = Normalize()
    Else
    max = "1"
  End If

  nn.MoveFirst
```

```
Do Until nn.EOF
   Select Case In_fkt                                    'Selektion der Eingabefunktion
   Case "linear"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format((nn!aid / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!lid / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn![Läsion Volumen] / max), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!aep / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!msep / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((pat!tsep / o_faktor), "###0.00000"), ",", ".")
   Case "Kehrwert"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / (nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((1 /
        ((nn![Läsion Volumen] / max) + i_faktor)), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        (pat!aep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / (pat!msep + o_faktor),
        "###0.00000"), ",", "."); " "; strReplace(Format(1 / (pat!tsep + o_faktor), "###0.00000"), ",", ".")
   Case "Logarithmus"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Log(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!lid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!msep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(pat!tsep + o_faktor), "###0.00000"), ",", ".")
   Case "Exponentiell"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Exp(-nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn![Läsion Volumen] / max)), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-pat!aep),
        "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-pat!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-pat!tsep), "###0.00000"), ",", ".")
   Case "TangensH"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(TanH(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!lid), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn![Läsion Volumen] / max), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(pat!tsep), "###0.00000"), ",", ".")
   Case "Sinus"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(Sin(nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn![Läsion
        Volumen] / max)), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(pat!aep), "###0.00000"),
        ",", "."); " "; strReplace(Format(Sin(pat!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(pat!tsep), "###0.00000"), ",", ".")
   Case "Wurzel"
        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)
        Print #fn, strReplace(Format(1 / Sqr(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        Sqr((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        Sqr(pat!aep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(pat!msep +
```

```
        o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(pat!tsep + o_faktor),
        "###0.00000"), ",", ".")
    End Select
    nn.MoveNext
  Loop
  pat.Close
  nn.Close
  Close fn

End Function

Public Function ep_gos_SNNS(i As String, o As String, file As String, i_faktor As String, o_faktor
As String, In_fkt As String, Out_fkt As String)
  'Function erzeugt Pattern-File für SNNSv4.1 Konstante: Datei, input/output units
  'Formal: 3xInput
  '       3xOutput

  'i              = Anzahl der Input Variablen
  'o              = Anzahl der Output Variablen
  'file           = Zielverzeichnis & Zieldatei
  'i_faktor       = Konstante zur Funktionsbestimmung im Eingaberaum
  'o_faktor       = Konstante zur Funktionsbestimmung im Ausgaberaum
  'In_fkt         = Schlüssel der Input-Funktion
  'Out_fkt        = Schlüssel der Output-Funktion

  Dim db As Database
  Dim nn As Recordset
  Dim dat As Recordset
  Dim pat As Recordset
  Dim sql, datei
  Set db = CurrentDb()
  Set nn = db.OpenRecordset("Patient Neu")

  datei = file
  Dim fn As Integer
  fn = FreeFile
  Open datei For Output As #fn                          'Öffnen der Ausgabedatei
                                                        'Dateikopf für den SNNS Patternfile
  Print #fn, "SNNS pattern definition file V3.2"
  Print #fn, "generated at " & Format(Date, "Long Date")
  Print #fn, ""
  Print #fn, "No. of patterns : " & DCount("pid", "Patient Neu")
  Print #fn, "No. of input units : " & i
  Print #fn, "No. of output units : " & o
  Print #fn, ""

  nn.MoveFirst
  Do Until nn.EOF
    Select Case In_fkt                                  'Selektion der Eingabefunktion
    Case "linear"
        Print #fn, strReplace(Format((nn!aep / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!msep / i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!tsep
        / i_faktor), "###0.00000"), ",", ".")

    Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!tsep + i_faktor), "###0.00000"), ",", ".")
    Case "Logarithmus"
```

```
        Print #fn, strReplace(Format(Log(nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!tsep + i_faktor), "###0.00000"), ",", ".")
    Case "Exponentiell"
        Print #fn, strReplace(Format(Exp(-nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn!tsep)), "###0.00000"), ",", ".")
    Case "TangensH"
        Print #fn, strReplace(Format(TanH(nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!tsep), "###0.00000"), ",", ".")
    Case "Sinus"
        Print #fn, strReplace(Format(Sin(nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn!tsep)),
        "###0.00000"), ",", ".")
    Case "Wurzel"
        Print #fn, strReplace(Format(1 / Sqr(nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!tsep + i_faktor), "###0.00000"), ",", ".")
    End Select

    Select Case Out_fkt                                    'Selektion der Ausgabefunktion
    Case "linear"
        Print #fn, strReplace(Format((nn!gose / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos3m / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos12m / o_faktor), "###0.00000"), ",", ".")
    Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!gos3m + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        (nn!gos12m + o_faktor), "###0.00000"), ",", ".")
    Case "Logarithmus"
        Print #fn, strReplace(Format(Log(nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!gos12m + o_faktor), "###0.00000"), ",", ".")
    Case "Exponetiell"
        Print #fn, strReplace(Format(Exp(-nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        nn!gos12m), "###0.00000"), ",", ".")
    Case "TangensH"
        Print #fn, strReplace(Format(TanH(nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!gos12m), "###0.00000"), ",", ".")
    Case "Sinus"
        Print #fn, strReplace(Format(Sin(nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!gos12m), "###0.00000"), ",", ".")
    Case "Wurzel"
        Print #fn, strReplace(Format(1 / Sqr(nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!gos12m + o_faktor), "###0.00000"), ",", ".")
    End Select
    nn.MoveNext
  Loop
  nn.Close
  Close fn

End Function
```

Public Function ep_gos_pascal(i As String, o As String, file As String, i_faktor As String, o_faktor As String, In_fkt As String, Out_fkt As String)

```
'Function erzeugt Pattern-File für Turbo Pascal Konstante: Datei, input/output units
'Formal: 3xInput 3xOutput

'i              = Anzahl der Input Variablen
'o              = Anzahl der Output Variablen
'file           = Zielverzeichnis & Zieldatei
'i_faktor       = Konstante zur Funktionsbestimmung im Eingaberaum
'o_faktor       = Konstante zur Funktionsbestimmung im Ausgaberaum
'In_fkt         = Schlüssel der Input-Funktion
'Out_fkt        = Schlüssel der Output-Funktion

Dim db As Database
Dim nn As Recordset
Dim dat As Recordset
Dim pat As Recordset
Dim sql, datei
Set db = CurrentDb()
Set nn = db.OpenRecordset("Patient Neu")

datei = file
Dim fn As Integer
fn = FreeFile
Open datei For Output As #fn                          'Öffnen der Ausgabedatei
                                                      'Kopfzeile des Patternfiles von Pascal
Print #fn, Trim(DCount("pid", "Patient Neu") / 2)      'Anzahl der Eingabedatensätze
Print #fn, i
Print #fn, Trim(DCount("pid", "Patient Neu") / 2)
Print #fn, o

nn.MoveFirst
Do Until nn.EOF
   Select Case In_fkt                                 'Selektion der Eingabefunktion
   Case "linear"
        Print #fn, strReplace(Format((nn!aep / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!msep / i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!tsep
        / i_faktor), "###0.00000"), ",", ".")
        Print #fn, strReplace(Format((nn!gose / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos3m / o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos12m / o_faktor), "###0.00000"), ",", ".")
   Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!tsep + i_faktor), "###0.00000"), ",", ".")
        Print #fn, strReplace(Format(1 / (nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!gos3m + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        (nn!gos12m + o_faktor), "###0.00000"), ",", ".")
   Case "Logarithmus"
        Print #fn, strReplace(Format(Log(nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!tsep + i_faktor), "###0.00000"), ",", ".")
        Print #fn, strReplace(Format(Log(nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!gos12m + o_faktor), "###0.00000"), ",", ".")
   Case "Exponentiell"
        Print #fn, strReplace(Format(Exp(-nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn!tsep)), "###0.00000"), ",", ".")
        Print #fn, strReplace(Format(Exp(-nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        nn!gos12m), "###0.00000"), ",", ".")
```

```
    Case "TangensH"
        Print #fn, strReplace(Format(TanH(nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!tsep), "###0.00000"), ",", ".")
        Print #fn, strReplace(Format(TanH(nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!gos12m), "###0.00000"), ",", ".")
    Case "Sinus"
        Print #fn, strReplace(Format(Sin(nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn!tsep)),
        "###0.00000"), ",", ".")
        Print #fn, strReplace(Format(Sin(nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!gos12m), "###0.00000"), ",", ".")
    Case "Wurzel"
        Print #fn, strReplace(Format(1 / Sqr(nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!tsep + i_faktor), "###0.00000"), ",", ".")
        Print #fn, strReplace(Format(1 / Sqr(nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!gos12m + o_faktor), "###0.00000"), ",", ".")
    End Select
    nn.MoveNext
  Loop
  nn.Close
  Close fn
End Function

Public Function ep_gos_havbpett(i As String, o As String, file As String, i_faktor As String,
o_faktor As String, In_fkt As String, Out_fkt As String)
  'Function erzeugt Pattern-File für Havbpett Konstante: Datei, input/output units
  'Formal: 3xInput 3xOutput

  'i          = Anzahl der Input Variablen
  'o          = Anzahl der Output Variablen
  'file       = Zielverzeichnis & Zieldatei
  'i_faktor   = Konstante zur Funktionsbestimmung im Eingaberaum
  'o_faktor   = Konstante zur Funktionsbestimmung im Ausgaberaum
  'In_fkt     = Schlüssel der Input-Funktion
  'Out_fkt    = Schlüssel der Output-Funktion

  Dim db As Database
  Dim nn As Recordset
  Dim dat As Recordset
  Dim pat As Recordset
  Dim sql, datei
  Set db = CurrentDb()
  Set nn = db.OpenRecordset("Patient Neu")

  datei = file
  Dim fn As Integer
  fn = FreeFile
  Open datei For Output As #fn                    'Öffnen der Ausgabedatei
                                                  'Kopfzeile des Patternfiles
  Print #fn, DCount("pid", "Patient Neu") / 2 & " " & i & " " & o

  nn.MoveFirst
  Do Until nn.EOF
    Select Case In_fkt
    Case "linear"
```

```
            Print #fn, strReplace(Format((nn!aep / i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((nn!msep / i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!tsep
            / i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!gose / o_faktor), "###0.00000"),
            ",", "."); " "; strReplace(Format((nn!gos3m / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((nn!gos12m / o_faktor), "###0.00000"), ",", ".")
        Case "Kehrwert"
            Print #fn, strReplace(Format(1 / (nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / (nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((nn!tsep + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
            (nn!gose + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / (nn!gos3m + o_faktor),
            "###0.00000"), ",", "."); " "; strReplace(Format(1 / (nn!gos12m + o_faktor), "###0.00000"), ",",
            ".")
        Case "Logarithmus"
            Print #fn, strReplace(Format(Log(nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!tsep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!gose + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!gos12m + o_faktor), "###0.00000"), ",", ".")
        Case "Exponentiell"
            Print #fn, strReplace(Format(Exp(-nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            (nn!tsep)), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-nn!gose), "###0.00000"), ",",
            "."); " "; strReplace(Format(Exp(-nn!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-nn!gos12m), "###0.00000"), ",", ".")
        Case "TangensH"
            Print #fn, strReplace(Format(TanH(nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!msep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!tsep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!gos12m), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn!tsep)),
            "###0.00000"), ",", "."); " "; strReplace(Format(Sin(nn!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!gos12m), "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(nn!aep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!msep + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!tsep + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1
            / Sqr(nn!gose + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(nn!gos3m +
            o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(nn!gos12m + o_faktor),
            "###0.00000"), ",", ".")
        End Select
        nn.MoveNext
    Loop
    nn.Close
    Close fn

End Function

Public Function gos_ep_havbpett(i As String, o As String, file As String, i_faktor As String,
o_faktor As String, In_fkt As String, Out_fkt As String)
    'Function erzeugt Pattern-File für Havbpett Konstante: Datei, input/output units
    'Formal: 3xInput 3xOutput

    'i              = Anzahl der Input Variablen
    'o              = Anzahl der Output Variablen
```

```
'file          = Zielverzeichnis & Zieldatei
'i_faktor      = Konstante zur Funktionsbestimmung im Eingaberaum
'o_faktor      = Konstante zur Funktionsbestimmung im Ausgaberaum
'In_fkt        = Schlüssel der Input-Funktion
'Out_fkt       = Schlüssel der Output-Funktion

Dim db As Database
Dim nn As Recordset
Dim dat As Recordset
Dim pat As Recordset
Dim sql, datei
Set db = CurrentDb()
Set nn = db.OpenRecordset("Patient Neu")

datei = file
Dim fn As Integer
fn = FreeFile
Open datei For Output As #fn                    'Öffnen der Ausgabedatei
                                                'Kopfzeile des Patternfiles
Print #fn, DCount("pid", "Patient Neu") / 2 & " " & i & " " & o

nn.MoveFirst
Do Until nn.EOF
   Select Case In_fkt
   Case "linear"
        Print #fn, strReplace(Format((nn!gose / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos3m / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos12m / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!aep / o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!msep
        / o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!tsep / o_faktor), "###0.00000"),
        ",", ".")
   Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos12m + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
        (nn!aep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / (nn!msep + o_faktor),
        "###0.00000"), ",", "."); " "; strReplace(Format(1 / (nn!tsep + o_faktor), "###0.00000"), ",", ".")
   Case "Logarithmus"
        Print #fn, strReplace(Format(Log(nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!gos12m + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!msep + o_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!tsep + o_faktor), "###0.00000"), ",", ".")
   Case "Exponentiell"
        Print #fn, strReplace(Format(Exp(-nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn!gos12m)), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-nn!aep), "###0.00000"), ",",
        "."); " "; strReplace(Format(Exp(-nn!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!tsep), "###0.00000"), ",", ".")
   Case "TangensH"
        Print #fn, strReplace(Format(TanH(nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!gos3m), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!gos12m), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!aep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!msep), "###0.00000"), ",", "."); " ";
        strReplace(Format(TanH(nn!tsep), "###0.00000"), ",", ".")
   Case "Sinus"
        Print #fn, strReplace(Format(Sin(nn!gose), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!gos3m), "###0.00000"), ",", "."); " ";
```

```
        strReplace(Format(Sin((nn!gos12m)), "###0.00000"), ",", "."); " ";
        strReplace(Format(Sin(nn!aep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(nn!msep),
        "###0.00000"), ",", "."); " "; strReplace(Format(Sin(nn!tsep), "###0.00000"), ",", ".")
    Case "Wurzel"
        Print #fn, strReplace(Format(1 / Sqr(nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!gos12m + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / Sqr(nn!aep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1
        / Sqr(nn!msep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 / Sqr(nn!tsep +
        o_faktor), "###0.00000"), ",", ".")
    End Select
    nn.MoveNext
  Loop
  nn.Close
  Close fn

End Function

Public Function gos_ep_pascal(i As String, o As String, file As String, i_faktor As String, o_faktor
As String, In_fkt As String, Out_fkt As String)
  'Function erzeugt Pattern-File für Turbo Pascal Konstante: Datei, input/output units
  'Formal: 3xInput 3xOutput

  'i            = Anzahl der Input Variablen
  'o            = Anzahl der Output Variablen
  'file         = Zielverzeichnis & Zieldatei
  'i_faktor     = Konstante zur Funktionsbestimmung im Eingaberaum
  'o_faktor     = Konstante zur Funktionsbestimmung im Ausgaberaum
  'In_fkt       = Schlüssel der Input-Funktion
  'Out_fkt      = Schlüssel der Output-Funktion

  Dim db As Database
  Dim nn As Recordset
  Dim dat As Recordset
  Dim pat As Recordset
  Dim sql, datei
  Set db = CurrentDb()
  Set nn = db.OpenRecordset("Patient Neu")

  datei = file
  Dim fn As Integer
  fn = FreeFile
  Open datei For Output As #fn                          'Öffnen der Ausgabedatei
                                                        'Kopfzeile des Patternfiles von Pascal
  Print #fn, Trim(DCount("pid", "Patient Neu") / 2)     'Anzahl der Eingabedatensätze
  Print #fn, i
  Print #fn, Trim(DCount("pid", "Patient Neu") / 2)
  Print #fn, o

  nn.MoveFirst
  Do Until nn.EOF
    Select Case In_fkt                                  'Selektion der Eingabefunktion
    Case "linear"
        Print #fn, strReplace(Format((nn!gose / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos3m / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!gos12m / i_faktor), "###0.00000"), ",", ".")
       Print #fn, strReplace(Format((nn!aep / o_faktor), "###0.00000"), ",", "."); " ";
       strReplace(Format((nn!msep / o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((nn!tsep
       / o_faktor), "###0.00000"), ",", ".")
    Case "Kehrwert"
```

```
            Print #fn, strReplace(Format(1 / (nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / (nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((nn!gos12m + i_faktor), "###0.00000"), ",", ".")
            Print #fn, strReplace(Format(1 / (nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / (nn!msep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
            (nn!tsep + o_faktor), "###0.00000"), ",", ".")
        Case "Logarithmus"
            Print #fn, strReplace(Format(Log(nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!gos12m + i_faktor), "###0.00000"), ",", ".")
            Print #fn, strReplace(Format(Log(nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!msep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!tsep + o_faktor), "###0.00000"), ",", ".")
        Case "Exponentiell"
            Print #fn, strReplace(Format(Exp(-nn!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-nn!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            (nn!gos12m)), "###0.00000"), ",", ".")
            Print #fn, strReplace(Format(Exp(-nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            nn!tsep), "###0.00000"), ",", ".")
        Case "TangensH"
            Print #fn, strReplace(Format(TanH(nn!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!gos12m), "###0.00000"), ",", ".")
            Print #fn, strReplace(Format(TanH(nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!msep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!tsep), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(nn!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin((nn!gos12m)), "###0.00000"), ",", ".")
            Print #fn, strReplace(Format(Sin(nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(nn!tsep),
            "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!gos12m + i_faktor), "###0.00000"), ",", ".")
            Print #fn, strReplace(Format(1 / Sqr(nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!msep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!tsep + o_faktor), "###0.00000"), ",", ".")
        End Select
        nn.MoveNext
    Loop
    nn.Close
    Close fn

End Function

Public Function gos_ep_SNNS(i As String, o As String, file As String, i_faktor As String, o_faktor
As String, In_fkt As String, Out_fkt As String)
    'Function erzeugt Pattern-File für SNNSv4.1 Konstante: Datei, input/output units
    'Formal: 3xInput
    '       3xOutput

    'i               = Anzahl der Input Variablen
    'o               = Anzahl der Output Variablen
    'file            = Zielverzeichnis & Zieldatei
    'i_faktor        = Konstante zur Funktionsbestimmung im Eingaberaum
    'o_faktor        = Konstante zur Funktionsbestimmung im Ausgaberaum
```

```
'In_fkt          = Schlüssel der Input-Funktion
'Out_fkt         = Schlüssel der Output-Funktion

Dim db As Database
Dim nn As Recordset
Dim dat As Recordset
Dim pat As Recordset
Dim sql, datei
Set db = CurrentDb()
Set nn = db.OpenRecordset("Patient Neu")

datei = file
Dim fn As Integer
fn = FreeFile
Open datei For Output As #fn                    'Öffnen der Ausgabedatei
                                                'Dateikopf für den SNNS Patternfile
Print #fn, "SNNS pattern definition file V3.2"
Print #fn, "generated at " & Format(Date, "Long Date")
Print #fn, ""
Print #fn, "No. of patterns : " & DCount("pid", "Patient Neu")
Print #fn, "No. of input units : " & i
Print #fn, "No. of output units : " & o
Print #fn, ""

nn.MoveFirst
Do Until nn.EOF
   Select Case In_fkt                           'Selektion der Eingabefunktion


   Case "linear"
      Print #fn, strReplace(Format((nn!gose / i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format((nn!gos3m / i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format((nn!gos12m / i_faktor), "###0.00000"), ",", ".")
   Case "Kehrwert"
      Print #fn, strReplace(Format(1 / (nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format(1 / (nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format((nn!gos12m + i_faktor), "###0.00000"), ",", ".")
   Case "Logarithmus"
      Print #fn, strReplace(Format(Log(nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format(Log(nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format(Log(nn!gos12m + i_faktor), "###0.00000"), ",", ".")
   Case "Exponentiell"
      Print #fn, strReplace(Format(Exp(-nn!gose), "###0.00000"), ",", "."); " ";
      strReplace(Format(Exp(-nn!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
      (nn!gos12m)), "###0.00000"), ",", ".")
   Case "TangensH"
      Print #fn, strReplace(Format(TanH(nn!gose), "###0.00000"), ",", "."); " ";
      strReplace(Format(TanH(nn!gos3m), "###0.00000"), ",", "."); " ";
      strReplace(Format(TanH(nn!gos12m), "###0.00000"), ",", ".")
   Case "Sinus"
      Print #fn, strReplace(Format(Sin(nn!gose), "###0.00000"), ",", "."); " ";
      strReplace(Format(Sin(nn!gos3m), "###0.00000"), ",", "."); " ";
      strReplace(Format(Sin((nn!gos12m)), "###0.00000"), ",", ".")
   Case "Wurzel"
      Print #fn, strReplace(Format(1 / Sqr(nn!gose + i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format(1 / Sqr(nn!gos3m + i_faktor), "###0.00000"), ",", "."); " ";
      strReplace(Format(1 / Sqr(nn!gos12m + i_faktor), "###0.00000"), ",", ".")
   End Select

   Select Case Out_fkt                          'Selektion der Ausgabefunktion
```

```
        Case "linear"
            Print #fn, strReplace(Format((nn!aep / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((nn!msep / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((nn!tsep / o_faktor), "###0.00000"), ",", ".")
        Case "Kehrwert"
            Print #fn, strReplace(Format(1 / (nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / (nn!msep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
            (nn!tsep + o_faktor), "###0.00000"), ",", ".")
        Case "Logarithmus"
            Print #fn, strReplace(Format(Log(nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!msep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(nn!tsep + o_faktor), "###0.00000"), ",", ".")
        Case "Exponetiell"
            Print #fn, strReplace(Format(Exp(-nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            nn!tsep), "###0.00000"), ",", ".")
        Case "TangensH"
            Print #fn, strReplace(Format(TanH(nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!msep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!tsep), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(nn!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(nn!tsep),
            "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(nn!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!msep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!tsep + o_faktor), "###0.00000"), ",", ".")
        End Select
        nn.MoveNext
    Loop
    nn.Close
    Close fn
End Function

Public Function gos_reduzierung()
    'Mit dieser Funktion werden die Skalen der Eingabewerte des GOS von dem Bereich 1-5
    ' auf 1-3 verkürzt wobei folgende Konvertierung gilt:
    '
    '1 = {1;2}
    '2 = {3]
    '3 = {4;5}

    DoCmd.SetWarnings False
    Dim db As Database
    Dim pat As Recordset
    Set db = CurrentDb()
    Set pat = db.OpenRecordset("Patient Neu")
    pat.MoveFirst
    Do Until pat.EOF
        pat.Edit
        Select Case pat!gose                                    'Anpassung des Wertes bei Entlassung
        Case 1, 2
            pat!gose = "1"
        Case 3
            pat!gose = "2"
        Case 4, 5
            pat!gose = "3"
        End Select
```

```
    Select Case pat!gos3m                          'Anpassung des Wertes nach 3 Monaten
    Case 1, 2
       pat!gos3m = "1"
    Case 3
       pat!gos3m = "2"
    Case 4, 5
       pat!gos3m = "3"
    End Select

    Select Case pat!gos12m                         'Anpassung des Wertes nach 12 Monaten
    Case 1, 2
       pat!gos12m = "1"
    Case 3
       pat!gos12m = "2"
    Case 4, 5
       pat!gos12m = "3"
    End Select
    pat.Update                                     'Datenaktualisierung
    pat.MoveNext
  Loop
  pat.Close
End Function
```

```
Public Function Pat_GOS_SNNS(i As String, o As String, file As String, i_faktor As String,
o_faktor As String, In_fkt As String, Out_fkt As String)
   'Function erzeugt Pattern-File für SNNSv4.1 Konstante: Datei, input/output units
   'Formal: 3xInput
   '       3xOutput

   'i              = Anzahl der Input Variablen
   'o              = Anzahl der Output Variablen
   'file           = Zielverzeichnis & Zieldatei
   'i_faktor       = Konstante zur Funktionsbestimmung im Eingaberaum
   'o_faktor       = Konstante zur Funktionsbestimmung im Ausgaberaum
   'In_fkt         = Schlüssel der Input-Funktion
   'Out_fkt        = Schlüssel der Output-Funktion

   Dim db As Database
   Dim nn As Recordset
   Dim dat As Recordset
   Dim pat As Recordset
   Dim sql, datei, max
   Set db = CurrentDb()
   Set nn = db.OpenRecordset("Läsionen Neu")

   datei = file
   Dim fn As Integer
   fn = FreeFile
   Open datei For Output As #fn                              'Öffnen der Ausgabedatei
                                                             'Dateikopf des SNNS files
   Print #fn, "SNNS pattern definition file V3.2"
   Print #fn, "generated at " & Format(Date, "Long Date")
   Print #fn, ""
   Print #fn, "No. of patterns : " & DCount("aid", "Läsionen Neu")
   Print #fn, "No. of input units : " & i
   Print #fn, "No. of output units : " & o
   Print #fn, ""

   If Forms![Pattern]![Normal] = True Then                   'Volumennormaisierung
      max = Normalize()
      Else
      max = "1"
   End If

   nn.MoveFirst
   Do Until nn.EOF
      Select Case In_fkt                                     'Selektion der Eingabefunktion
      Case "linear"
           Print #fn, strReplace(Format(Trim(Str(nn!aid) / i_faktor), "###0.00000"), ",", "."); " ";
           strReplace(Format(Trim(Str(nn!lid) / i_faktor), "###0.00000"), ",", "."); " ";
           strReplace(Format((nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
      Case "Kehrwert"
           Print #fn, strReplace(Format(1 / (nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
           strReplace(Format(1 / (nn!lid + 1), "###0.00000"), ",", "."); " "; strReplace(Format((1 /
           ((nn![Läsion Volumen] / max) + 1)), "###0.00000"), ",", ".")
      Case "Logarithmus"
           Print #fn, strReplace(Format(Log(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
           strReplace(Format(Log(nn!lid + i_faktor), "###0.00000"), ",", "."); " ";
           strReplace(Format(Log((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")


      Case "Exponentiell"
```

```
            Print #fn, strReplace(Format(Exp(-nn!aid), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            (nn![Läsion Volumen] / max)), "###0.00000"), ",", ".")
        Case "TangensH"
            Print #fn, strReplace(Format(TanH(nn!aid), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!lid), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(nn!aid), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn![Läsion
            Volumen] / max)), "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
            Sqr((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")
        End Select

        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)

        Select Case Out_fkt                                    'Selektion der Ausgabefunktion
        Case "linear"
            Print #fn, strReplace(Format((pat!gose / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((pat!gos3m / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((pat!gos12m / o_faktor), "###0.00000"), ",", ".")
        Case "Kehrwert"
            Print #fn, strReplace(Format(1 / (pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / (pat!gos3m + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1
            / (pat!gos12m + o_faktor), "###0.00000"), ",", ".")
        Case "Logarithmus"
            Print #fn, strReplace(Format(Log(pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(pat!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(pat!gos12m + o_faktor), "###0.00000"), ",", ".")
        Case "Exponentiell"
            Print #fn, strReplace(Format(Exp(-pat!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-pat!gos3m), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            pat!gos12m), "###0.00000"), ",", ".")
        Case "TangensH"
            Print #fn, strReplace(Format(TanH(pat!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(pat!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(pat!gos12m), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(pat!gose), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(pat!gos3m), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(pat!gos12m), "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(pat!gose + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(pat!gos3m + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(pat!gos12m + o_faktor), "###0.00000"), ",", ".")
        End Select
        nn.MoveNext
    Loop
    pat.Close
    nn.Close
    Close fn
End Function
Public Function Pat_EP_SNNS(i As String, o As String, file As String, i_faktor As String, o_faktor
As String, In_fkt As String, Out_fkt As String)
    'Function erzeugt Pattern-File für SNNSv4.1 Konstante: Datei, input/output units, faktor
    'Formal: 3xInput
```

```
'       3xOutput
'
'i              = Anzahl der Input Variablen
'o              = Anzahl der Output Variablen
'file           = Zielverzeichnis & Zieldatei
'i_faktor       = Konstante zur Funktionsbestimmung im Eingaberaum
'o_faktor       = Konstante zur Funktionsbestimmung im Ausgaberaum
'In_fkt         = Schlüssel der Input-Funktion
'Out_fkt        = Schlüssel der Output-Funktion

Dim db As Database
Dim nn As Recordset
Dim dat As Recordset
Dim pat As Recordset
Dim sql, datei, max
Set db = CurrentDb()
Set nn = db.OpenRecordset("Läsionen Neu")

datei = file
Dim fn As Integer
fn = FreeFile
Open datei For Output As #fn                                    'Öffnen der Ausgabedatei

Print #fn, "SNNS pattern definition file V3.2"       'Dateikopf des Patternfiles SNNS
Print #fn, "generated at " & Format(Date, "Long Date")
Print #fn, ""
Print #fn, "No. of patterns : " & DCount("aid", "Läsionen Neu")
Print #fn, "No. of input units : " & i
Print #fn, "No. of output units : " & o
Print #fn, ""

If Forms![Pattern]![Normal] = True Then             'Volumennormalisierung
   max = Normalize()
   Else
   max = "1"
End If

nn.MoveFirst
Do Until nn.EOF
   Select Case In_fkt                                           'Selektion der Eingabefunktion
   Case "linear"
        Print #fn, strReplace(Format((nn!aid / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn!lid / i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format((nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
   Case "Kehrwert"
        Print #fn, strReplace(Format(1 / (nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(1 / (nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format((1 /
        ((nn![Läsion Volumen] / max) + i_faktor)), "###0.00000"), ",", ".")
   Case "Logarithmus"
        Print #fn, strReplace(Format(Log(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log(nn!lid + i_faktor), "###0.00000"), ",", "."); " ";
        strReplace(Format(Log((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")


    Case "Exponentiell"
        Print #fn, strReplace(Format(Exp(-nn!aid), "###0.00000"), ",", "."); " ";
        strReplace(Format(Exp(-nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
        (nn![Läsion Volumen] / max)), "###0.00000"), ",", ".")
   Case "TangensH"
```

```
            Print #fn, strReplace(Format(TanH(nn!aid), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn!lid), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(nn![Läsion Volumen] / max), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(nn!aid), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(nn!lid), "###0.00000"), ",", "."); " "; strReplace(Format(Sin((nn![Läsion
            Volumen] / max)), "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(nn!aid + i_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(nn!lid + i_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
            Sqr((nn![Läsion Volumen] / max) + i_faktor), "###0.00000"), ",", ".")
        End Select

        sql = "Select [Patient Neu].* from [Patient Neu] where PID = " & nn!PID
        Set pat = db.OpenRecordset(sql)

    Select Case Out_fkt                                    'Selektion der Ausgabefunktion
        Case "linear"
            Print #fn, strReplace(Format((pat!aep / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((pat!msep / o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format((pat!tsep / o_faktor), "###0.00000"), ",", ".")
        Case "Kehrwert"
            Print #fn, strReplace(Format(1 / (pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / (pat!msep + o_faktor), "###0.00000"), ",", "."); " "; strReplace(Format(1 /
            (pat!tsep + o_faktor), "###0.00000"), ",", ".")
        Case "Logarithmus"
            Print #fn, strReplace(Format(Log(pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(pat!msep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(Log(pat!tsep + o_faktor), "###0.00000"), ",", ".")
        Case "Exponentiell"
            Print #fn, strReplace(Format(Exp(-pat!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Exp(-pat!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Exp(-
            pat!tsep), "###0.00000"), ",", ".")
        Case "TangensH"
            Print #fn, strReplace(Format(TanH(pat!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(pat!msep), "###0.00000"), ",", "."); " ";
            strReplace(Format(TanH(pat!tsep), "###0.00000"), ",", ".")
        Case "Sinus"
            Print #fn, strReplace(Format(Sin(pat!aep), "###0.00000"), ",", "."); " ";
            strReplace(Format(Sin(pat!msep), "###0.00000"), ",", "."); " "; strReplace(Format(Sin(pat!tsep),
            "###0.00000"), ",", ".")
        Case "Wurzel"
            Print #fn, strReplace(Format(1 / Sqr(pat!aep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(pat!msep + o_faktor), "###0.00000"), ",", "."); " ";
            strReplace(Format(1 / Sqr(pat!tsep + o_faktor), "###0.00000"), ",", ".")
        End Select
        nn.MoveNext
    Loop
    nn.Close
    pat.Close
    Close fn

End Function
Public Function Normalize() As String
    'Diese Function übernimmt die Normalisierung des Volumens
    'Als Rückgabewert liefert sie das Maximale Volumen der Datei Läsionen Neu

    Dim db As Database
    Dim nn As Recordset
    Dim max
```

```
    Set db = CurrentDb()
    Set nn = db.OpenRecordset("Läsionen Neu")

    max = 0
    nn.MoveFirst
    Do Until nn.EOF
       If nn![Läsion Volumen] > max Then
          max = nn![Läsion Volumen]
       End If
       nn.MoveNext
    Loop
    nn.Close
    Normalize = max

End Function

Public Function TanH(zahl As String)
    'Function zur Berechnung des Tangens hyperbolicus gemäß der Formel

    TanH = (Exp(zahl) - Exp(-zahl)) / (Exp(zahl) + Exp(-zahl))
End Function

Public Function transform(zahl As String, fkt As String, faktor As String) As String
    'Function die als Ergebnis den konvertierten Wert gemäß der ausgewählten Formel zurückgibt

    'zahl          = Zu konvertierender Wert
    'fkt           = Schlüssel der Transformationsfunktion
    'faktor        = Transformationsfaktor zur Funktion

    Select Case fkt
       Case 1                                                    'Logarithmus
          transform = Format((Log(zahl + faktor)), "###0.000")
       Case 2                                                    'Exponentiell
          transform = Format(Exp(-zahl), "###0.000")
       Case 3                                                    'Kehrwert
          transform = Format((1 / (zahl + faktor)), "###0.000")
       Case 4                                                    'linear
          transform = Format((zahl / faktor), "###0.000")
       Case 5                                                    'TangensH
          transform = Format((TanH(zahl)), "###0.000")
       Case 6                                                    'Sinus
          transform = Format((Sin(zahl)), "###0.000")
       Case 7                                                    'TangensH
          transform = Format(1 / (Sqr(zahl + faktor)), "###0.000")

    End Select
End Function



Public Function Ergebnis_Konverter()
    'Diese Funktion führt zur Analyse der erzielten Ergebnisdateien eine Ersetzungsroutine aus,
    'durch die das englische Zahlenformat (.) ducrh das deutsche Format (,) ersetzt wird

    Dim db As Database
    Dim nn As Recordset
    Dim snns As Recordset

    Set db = CurrentDb()
```

```
    Set nn = db.OpenRecordset("Ergebnis")
    Set snns = db.OpenRecordset("Ergebnis snns")

    DoCmd.SetWarnings False
    snns.MoveFirst

    DoCmd.RunSQL ("Delete Ergebnis.* from Ergebnis")       'Löschen der temporären Ergebnisdatei

    Do Until snns.EOF
        nn.AddNew
        snns.MoveNext
        nn!in1 = strReplace(snns!Feld1, ".", ",")
        nn!in2 = strReplace(snns!Feld2, ".", ",")
        nn!in3 = strReplace(snns!Feld3, ".", ",")
        snns.MoveNext
        nn!soll1 = strReplace(snns!Feld1, ".", ",")
        nn!soll2 = strReplace(snns!Feld2, ".", ",")
        nn!soll3 = strReplace(snns!Feld3, ".", ",")
        snns.MoveNext
        nn!ist1 = strReplace(snns!Feld1, ".", ",")
        nn!ist2 = strReplace(snns!Feld2, ".", ",")
        nn!ist3 = strReplace(snns!Feld3, ".", ",")
        nn.Update
        snns.MoveNext
    Loop
    snns.Close
    nn.Close

End Function

Public Function Fehler()
    'Durch diese Funktion werden die Ergebnisse des SNNS Simulators analysiert
    'Als Ergebnis liefert sie den quadratischen Fehler

    Dim db As Database
    Dim nn As Recordset
    Dim diff1, diff2, diff3 As Double

    Set db = CurrentDb()
    Set nn = db.OpenRecordset("Ergebnis")

    DoCmd.SetWarnings False
    nn.MoveFirst
    Do Until nn.EOF
        nn.Edit
        nn!differenz1 = nn!soll1 - nn!ist1
        nn!qdiff1 = nn!differenz1 ^ 2
        nn!differenz2 = nn!soll2 - nn!ist2
        nn!qdiff2 = nn!differenz2 ^ 2
        nn!differenz3 = nn!soll3 - nn!ist3
        nn!qdiff3 = nn!differenz3 ^ 2
        nn.Update
        nn.MoveNext
    Loop
    nn.Close

End Function

Public Function getFunktion(fkt As String, i As String) As String
    'Diese Funktion erzeugt den anzuzeigenden Funtionsterm incl. der Transformation
```

```
    'und der errechneten Parameter

    'fkt     = Schlüssel der gewählten Funktion
    'i       = errechneter Koeefizient der Funktion

Select Case fkt
    Case 1                                                  'Logarithmus
        getFunktion = "LN(x" & i & " + " & konstante(i) & ")"
    Case 2                                                  'Exponentiell
        getFunktion = "EXP(-x" & i & ")"
    Case 3                                                  'Kehrwert
        getFunktion = "/(x" & i & " + " & konstante(i) & ")"
    Case 4                                                  'linear
        getFunktion = "x" & i & "/" & konstante(i)
    Case 5                                                  'TangensH
        getFunktion = "tanH(x" & i & ")"
    Case 6                                                  'Sinus
        getFunktion = "sin(x" & i & ")"
    Case 7                                                  'Wurzel
        getFunktion = "/SQR(x" & i & " + " & konstante(i) & ")"
End Select
End Function

Public Function konstante(i As String) As String
    'Mit dieser Funtion wird die Konstante des Feldes i der temporären Tabelle ermittelt
    '
    'i       = Feldnummer
    'Zurückgegeben wird die Konstante des Formulars

Select Case i
    Case 1
        konstante = Forms![statistik]![faktor1]
    Case 2
        konstante = Forms![statistik]![Faktor2]
    Case 3
        konstante = Forms![statistik]![Faktor3]
    Case 4
        konstante = Forms![statistik]![Faktor4]
    Case 5
        konstante = Forms![statistik]![Faktor5]
    Case 6
        konstante = Forms![statistik]![Faktor6]
    Case 7
        konstante = Forms![statistik]![Faktor7]
    Case 8
        konstante = Forms![statistik]![Faktor8]
    Case 9
        konstante = Forms![statistik]![Faktor9]
End Select
End Function

Public Function funktion(i As String) As String
    'Diese Funktion liefert die Funktion der Variablen i

    'i       = zu betrachtende Feldnummer der temporären Tabelle
    'Rückgabewert ist die selektierte Funktion des Feldes i
Dim fkt As String

Select Case i
    Case 1
```

```
      fkt = Forms![statistik]![Funktion1]
   Case 2
      fkt = Forms![statistik]![Funktion2]
   Case 3
      fkt = Forms![statistik]![Funktion3]
   Case 4
      fkt = Forms![statistik]![Funktion4]
   Case 5
      fkt = Forms![statistik]![Funktion5]
   Case 6
      fkt = Forms![statistik]![Funktion6]
   Case 7
      fkt = Forms![statistik]![Funktion7]
   Case 8
      fkt = Forms![statistik]![Funktion8]
   Case 9
      fkt = Forms![statistik]![Funktion9]
End Select
funktion = getFunktion(fkt, i)                    'Aufruf der Stringerzeugungs-Funktion
End Function

Public Function Outfunktion(fkt As String, i As String) As String
   'Diese Funktion liefert die Transformationsfunkton der Outputfunktion
   '
   'fkt    = Schlüssel der Transformationsfunktion
   'i      = errechneter Koeefinzient der Funktion

Select Case fkt
   Case 1
      Outfunktion = "LN(y + " & konstante(i) & ") = "
   Case 2
      Outfunktion = "EXP(-y) = "
   Case 3
      Outfunktion = "1/(y + " & konstante(i) & ") ="
   Case 4
      Outfunktion = "y/" & konstante(i) & " = "
   Case 5
      Outfunktion = "tanH(y) = "
   Case 6
      Outfunktion = "sin(y) = "
   Case 7
      Outfunktion = "1/SQR(y + " & konstante(i) & ") = "
End Select
End Function
```

14.2 Modul Datenübernahme

```
Option Compare Database
Option Explicit

Private Sub Befehl3_Click()
DoCmd.OpenForm "Statistik"                                    'Öffnen des Statistikformulars
End Sub

Private Sub Befehl4_Click()
   'Öffnet in Abhängigkeit von dem Suchvorhaben die entsprechende Maske
Dim x, y
Dim file As String
                                                              'Wenn zu analysieren ist
If MsgBox("Möchten Sie die Analyse starten?", vbYesNo) = vbYes Then
   file = InputBox("Bitte geben Sie Laufwerk,Verzeichnis und Ergebnisdatei an", "Dateiangabe",
"C:\TEST.TXT")
   DoCmd.SetWarnings False
   DoCmd.RunSQL "Delete [Ergebnis SNNS].* from [Ergebnis SNNS]"
   DoCmd.TransferText acImportDelim, "Ergebnis_SNNS", "Ergebnis SNNS", file
   x = Ergebnis_Konverter()
   MsgBox " Die Ergebnisdaten wurden erfolgreich konvertiert"
  y = Fehler()
  DoCmd.OpenForm "Varianz"
  Else
  DoCmd.OpenForm "Varianz_suche"  'Ansonsten Öffne Suchenmaske
End If
End Sub

Private Sub Läsionen_Click()
   'Die Läsionsdaten werdn imporiert und gemäß den Referenzen konvertiert

If MsgBox("Die Datenübernahme löscht alle Daten der aktuellen Datenbank. Fortfahren ?", vbYesNo) =
vbNo Then
   Exit Sub
End If
DoCmd.SetWarnings False
Dim sql

                                                              'Löschen der temporären Tabellen
sql = "DELETE [Läsionen Neu].* FROM [Läsionen Neu];"
DoCmd.RunSQL sql
sql = "DELETE [Läsion reduziert].* FROM [Läsion reduziert];"
DoCmd.RunSQL sql
sql = "DELETE [Läsion reduziert durch Lage].* FROM [Läsion reduziert durch Lage];"
DoCmd.RunSQL sql
sql = "DELETE [Läsion reduziert durch Art].* FROM [Läsion reduziert durch Art];"
DoCmd.RunSQL sql
                                                              'Abfragen zum Datenimport
DoCmd.OpenQuery "knn Läsion-Lage"
DoCmd.OpenQuery "knn Läsion-Art"
DoCmd.OpenQuery "knn Läsion-Reduzierung"
DoCmd.OpenQuery "knn Läsion-Normalize"
DoCmd.OpenQuery "knn Läsion-Volumen"
DoCmd.OpenQuery "knn Läsion-Neu"
MsgBox " Die Läsionendaten wurden erfolgreich übernommen"

End Sub
Private Sub Patient_Click()
   'Übernahme der Patientendaten
```

```
If MsgBox("Die Datenübernahme löscht alle Daten der aktuellen Datenbank. Fortfahren ?", vbYesNo) =
vbNo Then
   Exit Sub
End If
DoCmd.SetWarnings False
Dim x, sql
                                                    'Löschen der temporären Patiententabelle
sql = "DELETE [Patient Neu].* FROM [Patient Neu];"
DoCmd.RunSQL sql
                                                    'Abfrage zum Datenimport
DoCmd.OpenQuery "knn Patient", acViewNormal
If MsgBox("Wolllen Sie die Daten reduziert übernehmen ?", vbYesNo) = vbYes Then
   x = gos_reduzierung()                                    'Skalenreduzierung
End If
MsgBox " Die Patientendaten wurden erfolgreich übernommen"
End Sub


Private Sub Pattern_Click()
   'Öffnen der Patternerzeugungsmaske
DoCmd.OpenForm "Pattern"
End Sub
```

14.3 Modul Patternerzeugung

```
Option Compare Database
Option Explicit

Private Sub Befehl6_Click()
   'Erzeugung der Patternfiles in Abhängigkeit von der Zuordnung der Transformationsfunkiton
   'und des gewählten Simulators

   Dim x
                                                    'Fehlerprüfung bei fehlenden Angaben
   If IsNull(Me!i) Then
      MsgBox "Bitte Anzahl der Input Variablen eingeben"
      Exit Sub
   End If
   If IsNull(Me!o) Then
      MsgBox "Bitte Anzahl der Output Variablen eingeben"
      Exit Sub
   End If
   If IsNull(Me!file) Then
      MsgBox "Bitte Zielverzeichnis & Zieldatei eingeben"
      Exit Sub
   End If
                                                'Selektion der Konvertierungsfunktion
DoCmd.Hourglass True
If Me!pat_gos = True Then
   If Me!snns = True Then
      x = Pat_GOS_SNNS(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
      Else
      If Me!tp = True Then
         x = Pat_gos_Pascal(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
         Else
         If Me!havbpett = True Then
            x = Pat_gos_Havbpett(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
            Else
            MsgBox " Bitte einen Simulator wählen"
            Exit Sub
         End If
      End If
   End If
End If
If Me!pat_ep = True Then
   If Me!snns = True Then
      x = Pat_EP_SNNS(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
      Else
      If Me!tp = True Then
         x = Pat_ep_Pascal(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
         Else
         If Me!havbpett = True Then
            x = Pat_ep_havbpett(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
            Else
            MsgBox " Bitte einen Simulator wählen"
            Exit Sub
         End If
      End If
   End If
End If
If Me!gos_ep = True Then
   If Me!snns = True Then
```

```
        x = gos_ep_SNNS(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
        Else
        If Me!tp = True Then
           x = gos_ep_pascal(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
           Else
           If Me!havbpett = True Then
              x = gos_ep_havbpett(i, o, file, i_faktor, o_faktor, In_fkt, Out_fkt)
              Else
              MsgBox " Bitte einen Simulator wählen"
              Exit Sub
           End If
        End If
     End If
End If
DoCmd.Hourglass False
MsgBox " Das Patternfile wurde erfolgreich konvertiert"

End Sub



Private Sub Form_Load()
   'Einstellung der geladenen Felder beim Öffnen der Maske

Me!In_fkt = "Linear"
Me!Out_fkt = "Linear"
Me!file = "C:\test.txt"
Me!i = "3"
Me!o = "3"
Me!pat_gos = False
Me!pat_ep = False
Me!gos_ep = False
Me!snns = False
Me!tp = False
Me!havbpett = False
Me!i_faktor.Visible = True
Me!o_faktor.Visible = True
End Sub


Private Sub gos_ep_Click()
   'Switchschalter für die gewälten Zuordnungen Simulator-Abhängigkeit

Me!ep_gos = False
Me!pat_gos = False
Me!pat_ep = False
Me!Normal.Visible = False
End Sub

Private Sub havbpett_Click()
   'Switchschalter für die gewälten Zuordnungen Simulator-Abhängigkeit

Me!snns = False
Me!tp = False
End Sub


Private Sub In_fkt_AfterUpdate()
   'Schalter für Funktionen, die eine Konstante besitzen
```

```
If Me!In_fkt = "Linear" Or In_fkt = "Kehrwert" Or In_fkt = "Wurzel" Or In_fkt = "Logarithmus" Then
  Me!i_faktor.Visible = True
  Else
  Me!i_faktor.Visible = False
End If
End Sub

Private Sub Out_fkt_AfterUpdate()
   'Schalter für Funktionen, die eine Konstante besitzen

If Me!Out_fkt = "Linear" Or Out_fkt = "Kehrwert" Or Out_fkt = "Wurzel" Or Out_fkt = "Logarithmus"
Then
  Me!o_faktor.Visible = True
  Else
  Me!o_faktor.Visible = False
End If
End Sub

Private Sub pat_ep_Click()
   'Switchschalter für die gewälten Zuordnungen Simulator-Abhängigkeit

Me!ep_gos = False
Me!pat_gos = False
Me!gos_ep = False
Me!Normal.Visible = True
End Sub

Private Sub pat_gos_Click()
   'Switchschalter für die gewälten Zuordnungen Simulator-Abhängigkeit

Me!ep_gos = False
Me!gos_ep = False
Me!pat_ep = False
Me!Normal.Visible = True
End Sub

Private Sub snns_Click()
   'Switchschalter für die gewälten Zuordnungen Simulator-Abhängigkeit

Me!tp = False
Me!havbpett = False
End Sub

Private Sub tp_Click()
   'Switchschalter für die gewälten Zuordnungen Simulator-Abhängigkeit

Me!snns = False
Me!havbpett = False
End Sub
```

14.4 Modul Statistik

```
Option Compare Database
Option Explicit


Private Sub Befehl118_Click()
    'Mit diesem Programm werden die Daten aus der Datenbank extrahiert und diese Daten
    'zur Berechnung nach Excel übergeben, um das Ergebnis wieder zurückzubekommen
    'Der Eingaberaum der Variablen wird dynamisch gehalten
    'Die Kommunikation der zuständigen Programme wird unter Verwendung von
    'Viasual Basic 5.0 realisiert

Dim db As Database
Dim pat As Recordset
Dim nn As Recordset
Dim s As Recordset
Dim sql, x, quelle
Dim str1
Dim i As Integer
Dim objExcel As Object
Dim objExcelWS As Object
Dim objExcelWB As Object
Dim strSpalte As String
Dim e, a As Integer
Dim Pearson30 As String
Dim Pearson320 As String
Dim Verz As String
Dim strControl As String
Dim z As Integer


Set db = CurrentDb()                                    'Öffnen der benötigten Tabellen
Set nn = db.OpenRecordset("Läsionen Neu")
Set pat = db.OpenRecordset("Patient Neu")
Set s = db.OpenRecordset("Datei")

Me!Regressionsfunktion.Visible = False
Me!Bestimmt.Visible = False
Me!Pearson.Visible = False

DoCmd.SetWarnings False                                 'Löschen der temporären Tabellen
DoCmd.RunSQL ("Delete Datei.* from Datei")
DoCmd.Hourglass True

Verz = "C:\KNN\"                                        'Verzeichnis der Excelsheets
Pearson30 = Verz & "PEARSON30.XLS"
Pearson320 = Verz & "PEARSON320.XLS"
i = 0
strControl = "Klasse"
For z = 1 To 9
   strControl = "Klasse" & Trim(Str(z))
   If Me(strControl) = "in" Then                        'Ermittlung der Anzahl von
Eingabevariablen
      i = i + 1
   End If
Next z
                                                        'Wenn Läsionsdaten Datenquelle sind
```

```
If (IsNull(Me!Klasse7) Or Me!Klasse7 = "nicht relevant") And (IsNull(Me!Klasse8) Or Me!Klasse8 =
"nicht relevant") And (IsNull(Me!Klasse9) Or Me!Klasse9 = "nicht relevant") Then
   pat.MoveFirst                                    'Patienten Neu wird initialisiert

   Do Until pat.EOF
      s.AddNew
      If Me!Klasse1 = "in" Then                     'Festlegung des Eingabewertes
         s!in1 = transform(pat!gose, Funktion1, faktor1)
         Else
         If Me!Klasse1 = "out" Then                 'Klassifizierung des Outputs
            s!out = transform(pat!gose, Funktion1, faktor1)   ‘Transformation
            str1 = Outfunktion(Me!Funktion1, 1)
         End If
      End If
      If Me!Klasse2 = "in" Then
         s!in2 = transform(pat!gos3m, Funktion2, Faktor2)
         Else
         If Me!Klasse2 = "out" Then
            s!out = transform(pat!gos3m, Funktion2, Faktor2)
            str1 = Outfunktion(Me!Funktion2, 2)
         End If
      End If
      If Me!Klasse3 = "in" Then
         s!in3 = transform(pat!gos12m, Funktion3, Faktor3)
         Else
         If Me!Klasse3 = "out" Then
            s!out = transform(pat!gos12m, Funktion3, Faktor3)
            str1 = Outfunktion(Me!Funktion3, 3)
         End If
      End If
      If Me!Klasse4 = "in" Then
         s!in4 = transform(pat!aep, Funktion4, Faktor4)
         Else
         If Me!Klasse4 = "out" Then
            s!out = transform(pat!aep, Funktion4, Faktor4)
            str1 = Outfunktion(Me!Funktion4, 4)
         End If
      End If
      If Me!Klasse5 = "in" Then
         s!in5 = transform(pat!msep, Funktion5, Faktor5)
         Else
         If Me!Klasse5 = "out" Then
            s!out = transform(pat!msep, Funktion5, Faktor5)
            str1 = Outfunktion(Me!Funktion5, 5)
         End If
      End If
      If Me!Klasse6 = "in" Then
         s!in6 = transform(pat!tsep, Funktion6, Faktor6)
         Else
         If Me!Klasse6 = "out" Then
            s!out = transform(pat!tsep, Funktion6, Faktor6)
            str1 = Outfunktion(Me!Funktion6, 6)
         End If
      End If
      s.Update
      pat.MoveNext
   Loop
   pat.Close
                                                    'Erzeugung des Excelobjekte
   Set objExcel = CreateObject("Excel.Application")
```

```
   objExcel.Workbooks.Open FileName:=Pearson30
   Set objExcelWS = objExcel.Sheets(Trim(Str(i)) & " Variable")

   a = 0
   For z = 1 To 9                                    'Tabellengenerierung der Eingaben
      strSpalte = "in" & Trim(Str(z))
      If blnSumme(strSpalte, s) = True Then          'Überprüfung des dynamischen Raums
         a = a + 1
         s.MoveFirst
         For e = 2 To 31                             'Füllen der Felder pro Spalte
            objExcelWS.cells(e, a).Value = s(strSpalte)
            s.MoveNext
         Next e
      End If
   Next z
   strSpalte = "out"
   s.MoveFirst

   For e = 2 To 31                                   'Generierung des Ausgaberaums
      objExcelWS.cells(e, i + 1).Value = s(strSpalte)
      s.MoveNext
   Next e

   If Me.alpha > 0 Then                              'Flag zur Generierung des
Konfidenzintervalls
      objExcelWS.cells(36, 16).Value = Me!alpha
      Me.Konfidenz.Visible = True
   End If
   quelle = Pearson30

   If i = 1 Then                                'Wenn lineare Zuordnung dann Grafikerzeugung
      Diagramm.Visible = True
      Set objExcelWS = objExcel.Sheets("Graphik")
      objExcelWS.chartobjects(1).Activate
      objExcelWS.chartobjects(1).copy
      Diagramm.Class = "Excel.Chart"
      Diagramm.OLETypeAllowed = acOLEEmbedded
      Diagramm.SourceDoc = Pearson30
      Diagramm.Action = acOLEPaste
      Diagramm.SizeMode = acOLESizeStretch
      Set objExcelWS = objExcel.Sheets(Trim(Str(i)) & " Variable")
      Else
      Diagramm.Visible = False                      'Sonst ist Grafik unsichtbar
   End If

   Select Case i                                'Erzeugung der Abbildungsfunktion
      Case 1                                          'in Abhängigkeit von der Inputanzahl
         For z = 1 To 9
            strSpalte = "in" & Trim(Str(z))
            If blnSumme(strSpalte, s) = True Then
               Me.Regressionsfunktion = str1 & Format(objExcelWS.cells(46, 18), "###0.000") & " " &
funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(47, 18), "###0.000")
            End If
         Next z
      Case 2
         a = 0
         For z = 1 To 9
            strSpalte = "in" & Trim(Str(z))
            If blnSumme(strSpalte, s) = True Then
               a = a + 1
```

```
            If a <> 2 Then
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + "
            Else
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(46 + a)), 18),
                "###0.000")
            End If
        End If
        Me.Regressionsfunktion = str1
    Next z
Case 3
    a = 0
    For z = 1 To 9
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
            a = a + 1
            If a <> 3 Then
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + "
            Else
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(46 + a)), 18),
                "###0.000")
            End If
        End If
        Me.Regressionsfunktion = str1
    Next z
Case 4
    a = 0
    For z = 1 To 9
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
            a = a + 1
            If a <> 4 Then
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + "
            Else
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(46 + a)), 18),
                "###0.000")
            End If
        End If
        Me.Regressionsfunktion = str1
    Next z
Case 5
    a = 0
    For z = 1 To 9
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
            a = a + 1
            If a <> 5 Then
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + "
            Else
                str1 = str1 & Format(objExcelWS.cells(Trim(Str(45 + a)), 18), "###0.000") & " " &
                funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(46 + a)), 18),
                "###0.000")
            End If
        End If
```

```
            Me.Regressionsfunktion = str1
        Next z
    End Select
                                                    'Übergabe von Betimmtheitsmaß und
                                                    'Korrelationskoeffizient
    Me.Bestimmt = Format(objExcelWS.cells(40, 16), "###0.000")
    Me.Pearson = Format(objExcelWS.cells(41, 16), "###0.000")

    If Me.alpha > 0 Then                            'Wenn Konfidenzintervall aktiviert ist
      Me.Konfidenz = Format(objExcelWS.cells(38, 16), "###0.000")
      Else
      Me.Konfidenz = "0.00"
    End If

    Else
'wenn Läsiondaten betroffen sind, äußere If-Schleife

    nn.MoveFirst
    Do Until nn.EOF
      s.AddNew                                      'Ermittlung der betroffenen Felder
      If Me!Klasse7 = "in" Then
        s!in7 = transform(nn!aid, Funktion7, Faktor7)
        Else
        If Me!Klasse7 = "out" Then
          s!out = transform(nn!aid, Funktion7, Faktor7)
          str1 = Outfunktion(Me!Funktion7, 7)
        End If
      End If
      If Me!Klasse8 = "in" Then
        s!in8 = transform(nn!lid, Funktion8, Faktor8)
        Else
        If Me!Klasse8 = "out" Then
          s!out = transform(nn!lid, Funktion8, Faktor8)
          str1 = Outfunktion(Me!Funktion8, 8)
        End If
      End If
      If Me!Klasse9 = "in" Then
        s!in9 = transform(nn![Läsion Volumen] / Normalize(), Funktion9, Faktor9)
        Else
        If Me!Klasse9 = "out" Then
          s!out = transform(nn![Läsion Volumen] / Normalize(), Funktion9, Faktor9)
          str1 = Outfunktion(Me!Funktion9, 9)
        End If
      End If

                                                    'Selektion der betroffenen Patientendaten
      sql = "Select distinctrow [Patient Neu].* from [Patient Neu] where [Patient Neu].PID =" & nn!PID
      Set pat = db.OpenRecordset(sql)

      If Me!Klasse1 = "in" Then
        s!in1 = transform(pat!gose, Funktion1, faktor1)
        Else
        If Me!Klasse1 = "out" Then
          s!out = transform(pat!gose, Funktion1, faktor1)
          str1 = Outfunktion(Me!Funktion1, 1)
        End If
      End If
      If Me!Klasse2 = "in" Then
        s!in2 = transform(pat!gos3m, Funktion2, Faktor2)
        Else
```

```
        If Me!Klasse2 = "out" Then
          s!out = transform(pat!gos3m, Funktion2, Faktor2)
          str1 = Outfunktion(Me!Funktion2, 2)
        End If
      End If
      If Me!Klasse3 = "in" Then
        s!in3 = transform(pat!gos12m, Funktion3, Faktor3)
        Else
        If Me!Klasse3 = "out" Then
          s!out = transform(pat!gos12m, Funktion3, Faktor3)
          str1 = Outfunktion(Me!Funktion3, 3)
        End If
      End If
      If Me!Klasse4 = "in" Then
        s!in4 = transform(pat!aep, Funktion4, Faktor4)
        Else
        If Me!Klasse4 = "out" Then
          s!out = transform(pat!aep, Funktion4, Faktor4)
          str1 = Outfunktion(Me!Funktion4, 4)
        End If
      End If
      If Me!Klasse5 = "in" Then
        s!in5 = transform(pat!msep, Funktion5, Faktor5)
        Else
        If Me!Klasse5 = "out" Then
          s!out = transform(pat!msep, Funktion5, Faktor5)
          str1 = Outfunktion(Me!Funktion5, 5)
        End If
      End If
      If Me!Klasse6 = "in" Then
        s!in6 = transform(pat!tsep, Funktion6, Faktor6)
        Else
        If Me!Klasse6 = "out" Then
          s!out = transform(pat!tsep, Funktion6, Faktor6)
          str1 = Outfunktion(Me!Funktion6, 6)
        End If
      End If
      s.Update
      nn.MoveNext
    Loop
    pat.Close
    nn.Close
                                                    'Dimensionierung der Excel-Objekte
    Set objExcel = CreateObject("Excel.Application")
    objExcel.Workbooks.Open FileName:=Pearson320
    Set objExcelWS = objExcel.Sheets(Trim(Str(i)) & " Variable")

    a = 0
    For z = 1 To 9                                  'Konvertierung des betroffenen
Eingaberaums
      strSpalte = "in" & Trim(Str(z))
      If blnSumme(strSpalte, s) = True Then
        a = a + 1
        s.MoveFirst
        For e = 2 To 321
          objExcelWS.cells(e, a).Value = s(strSpalte)
          s.MoveNext
        Next e
      End If
    Next z
```

```
   strSpalte = "out"
   s.MoveFirst

   For e = 2 To 321                                    'Konvertierung des Ausgaberaums
      objExcelWS.cells(e, i + 1).Value = s(strSpalte)
      s.MoveNext
   Next e

   If Me.alpha > 0 Then                                'Bestimmung des Konfidenzintervalls
      objExcelWS.cells(326, 16).Value = Me!alpha
      Me.Konfidenz.Visible = True
   End If
   quelle = Pearson320

   If i = 1 Then                                       'Diagrammberechnung wenn f(x)
      Diagramm.Visible = True
      Set objExcelWS = objExcel.Sheets("Graphik")
      objExcelWS.chartobjects(1).Activate
      objExcelWS.chartobjects(1).copy
      Diagramm.Class = "Excel.Chart"
      Diagramm.OLETypeAllowed = acOLEEmbedded
      Diagramm.SourceDoc = Pearson320
      Diagramm.Action = acOLEPaste
      Diagramm.SizeMode = acOLESizeStretch
      Set objExcelWS = objExcel.Sheets(Trim(Str(i)) & " Variable")
      Else
      Diagramm.Visible = False
   End If


'Erzeugung der Abbildungsfunkion incl. Transformation
   Select Case i
      Case 1
         For z = 1 To 9
            strSpalte = "in" & Trim(Str(z))
            If blnSumme(strSpalte, s) = True Then
                  Me.Regressionsfunktion = str1 & Format(objExcelWS.cells(336, 18), "###0.000") & " "
                  & funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(337, 18), "###0.000")
            End If
         Next z
      Case 2
         a = 0
         For z = 1 To 9
            strSpalte = "in" & Trim(Str(z))
            If blnSumme(strSpalte, s) = True Then
               a = a + 1
               If a <> 2 Then
                     str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                     funktion(Trim(Str(z))) & " + "
                  Else
                     str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                     funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
                     "###0.000")
               End If
            End If
            Me.Regressionsfunktion = str1
         Next z
      Case 3
         a = 0
         For z = 1 To 9
```

```
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
          a = a + 1
          If a <> 3 Then
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + "
            Else
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
              "###0.000")
          End If
        End If
        Me.Regressionsfunktion = str1
      Next z
    Case 4
      a = 0
      For z = 1 To 9
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
          a = a + 1
          If a <> i Then
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + "
            Else
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
              "###0.000")
          End If
        End If
        Me.Regressionsfunktion = str1
      Next z
    Case 5
      a = 0
      For z = 1 To 9
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
          a = a + 1
          If a <> i Then
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + "
            Else
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
              "###0.000")
          End If
        End If
        Me.Regressionsfunktion = str1
      Next z
    Case 6
      a = 0
      For z = 1 To 9
        strSpalte = "in" & Trim(Str(z))
        If blnSumme(strSpalte, s) = True Then
          a = a + 1
          If a <> i Then
              str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
              funktion(Trim(Str(z))) & " + "
            Else
```

```
                    str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                    funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
                    "###0.000")
                End If
            End If
            Me.Regressionsfunktion = str1
        Next z
    Case 7
        a = 0
        For z = 1 To 9
            strSpalte = "in" & Trim(Str(z))
            If blnSumme(strSpalte, s) = True Then
                a = a + 1
                If a <> i Then
                    str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                    funktion(Trim(Str(z))) & " + "
                Else
                    str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                    funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
                    "###0.000")
                End If
            End If
            Me.Regressionsfunktion = str1
        Next z
    Case 8
        a = 0
        For z = 1 To 9
            strSpalte = "in" & Trim(Str(z))
            If blnSumme(strSpalte, s) = True Then
                a = a + 1
                If a <> i Then
                    str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                    funktion(Trim(Str(z))) & " + "
                Else
                    str1 = str1 & Format(objExcelWS.cells(Trim(Str(335 + a)), 18), "###0.000") & " " &
                    funktion(Trim(Str(z))) & " + " & Format(objExcelWS.cells(Trim(Str(336 + a)), 18),
                    "###0.000")
                End If
            End If
            Me.Regressionsfunktion = str1
        Next z
End Select
'Übergabe von Bestimmtheitsmaß und Korrelationskoeffiezient
Me.Bestimmt = Format(objExcelWS.cells(330, 16), "###0.000")
Me.Pearson = Format(objExcelWS.cells(331, 16), "###0.000")

If Me.alpha > 0 Then                                    'Berechnung des Konfidenzintervalls
    Me.Konfidenz = Format(objExcelWS.cells(328, 16), "###0.000")
    Else
    Me.Konfidenz = "0.00"
End If
End If

DoCmd.Hourglass False
If quelle = Pearson30 Then                              'Schließen des aktivierten Excelsheets
    objExcel.Workbooks("Pearson30.XLS").Close (False)
    Else
    If quelle = Pearson320 Then
        objExcel.Workbooks("Pearson320.XLS").Close (False)
    End If
```

```
End If

objExcel.Quit
Set objExcelWS = Nothing                                    'Zurücksetzen der Objekte
Set objExcel = Nothing

MsgBox "Die Daten wurden erfolgreich analysiert"
s.Close
Me!Regressionsfunktion.Visible = True
Me!Bestimmt.Visible = True
Me!Pearson.Visible = True
Me.Regressionsfunktion.SetFocus
End Sub


Private Sub faktor1_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.faktor1 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.faktor1 = 1
End If
End Sub

Private Sub Faktor2_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor2 < 0 Or Me.Faktor2 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor2 = 1
End If
End Sub




Private Sub Faktor3_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor3 < 0 Or Me.Faktor3 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor3 = 1
End If
End Sub

Private Sub Faktor4_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor4 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor4 = 1
End If
End Sub


Private Sub Faktor5_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten
```

```
If Me.faktor1 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.faktor1 = 1
End If
End Sub

Private Sub Faktor6_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor6 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor6 = 1
End If
End Sub


Private Sub Faktor7_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor7 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor7 = 1
End If
End Sub


Private Sub Faktor8_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor8 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor8 = 1
End If
End Sub


Private Sub Faktor9_AfterUpdate()
   'Fehlerroutine zum Abfangen negativer Konstanten

If Me.Faktor9 < 0 Or Me.faktor1 = 0 Then
   MsgBox "Dieses Feld ist nur für Zahlen größer 0.00 zugelassen."
   Me.Faktor9 = 1
End If
End Sub


Private Sub Form_Load()
   'Beim Laden des Formulars werden die Ergebnisfelder auf unsichtbar geschaltet

Me!Regressionsfunktion.Visible = False
Me!Bestimmt.Visible = False
Me!Pearson.Visible = False
Me!Diagramm.Visible = False
Me!Konfidenz.Visible = False
End Sub

Private Sub Funktion1_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter
```

```
If Me!Funktion1 = 3 Or Me!Funktion1 = 4 Or Me!Funktion1 = 7 Or Me!Funktion1 = 1 Then
   Me!faktor1.Visible = True
   Else
   Me!faktor1.Visible = False
End If
End Sub

Private Sub Funktion2_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter
If Me!Funktion2 = 3 Or Me!Funktion2 = 4 Or Me!Funktion2 = 7 Or Me!Funktion2 = 1 Then
   Me!Faktor2.Visible = True
   Else
   Me!Faktor2.Visible = False
End If
End Sub

Private Sub Funktion3_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion3 = 3 Or Me!Funktion3 = 4 Or Me!Funktion3 = 7 Or Me!Funktion3 = 1 Then
   Me!Faktor3.Visible = True
   Else
   Me!Faktor3.Visible = False
End If
End Sub

Private Sub Funktion4_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion4 = 3 Or Me!Funktion4 = 4 Or Me!Funktion4 = 7 Or Me!Funktion4 = 1 Then
   Me!Faktor4.Visible = True
   Else
   Me!Faktor4.Visible = False
End If
End Sub


Private Sub Funktion5_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion5 = 3 Or Me!Funktion5 = 4 Or Me!Funktion5 = 7 Or Me!Funktion5 = 1 Then
   Me!Faktor5.Visible = True
   Else
   Me!Faktor5.Visible = False
End If
End Sub




Private Sub Funktion6_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion6 = 3 Or Me!Funktion6 = 4 Or Me!Funktion6 = 7 Or Me!Funktion6 = 1 Then
   Me!Faktor6.Visible = True
   Else
   Me!Faktor6.Visible = False
End If
```

```
End Sub

Private Sub Funktion7_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion7 = 3 Or Me!Funktion7 = 4 Or Me!Funktion7 = 7 Or Me!Funktion7 = 1 Then
   Me!Faktor7.Visible = True
   Else
   Me!Faktor7.Visible = False
End If
End Sub


Private Sub Funktion8_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion8 = 3 Or Me!Funktion8 = 4 Or Me!Funktion8 = 7 Or Me!Funktion8 = 1 Then
   Me!Faktor8.Visible = True
   Else
   Me!Faktor8.Visible = False
End If
End Sub

Private Sub Funktion9_AfterUpdate()
   'Sichtbarschalten bei Funktionen mit Transformationskonstanter

If Me!Funktion9 = 3 Or Me!Funktion9 = 4 Or Me!Funktion9 = 7 Or Me!Funktion9 = 1 Then
   Me!Faktor9.Visible = True
   Else
   Me!Faktor9.Visible = False
End If
End Sub

Private Sub Klasse1_AfterUpdate()
   'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse1 = "out" Then
   If Me!Klasse2 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse3 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse4 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse5 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse6 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
```

```
      Exit Sub
   End If
   If Me!Klasse7 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse8 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse9 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse1 = "nicht relevant"
      Exit Sub
   End If
End If

End Sub

Private Sub Klasse2_AfterUpdate()
   'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse2 = "out" Then
   If Me!Klasse1 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse3 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse4 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse5 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse6 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse7 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse8 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse2 = "nicht relevant"
      Exit Sub
   End If
```

```
    If Me!Klasse9 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse2 = "nicht relevant"
        Exit Sub
    End If
End If

End Sub

Private Sub Klasse3_AfterUpdate()
    'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse3 = "out" Then
    If Me!Klasse1 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse2 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse4 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse5 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse6 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse7 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse8 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse9 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse3 = "nicht relevant"
        Exit Sub
    End If
End If

End Sub

Private Sub Klasse4_AfterUpdate()
    'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde
```

```
If Me!Klasse4 = "out" Then
   If Me!Klasse1 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse2 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse3 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse5 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse6 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse7 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse8 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse9 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse4 = "nicht relevant"
      Exit Sub
   End If
End If
End Sub


Private Sub Klasse5_AfterUpdate()
   'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse5 = "out" Then
   If Me!Klasse1 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse5 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse2 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse5 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse3 = "out" Then
```

```
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse5 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse4 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse5 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse6 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse5 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse7 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse5 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse8 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse5 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse9 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse5 = "nicht relevant"
        Exit Sub
    End If
End If
End Sub

Private Sub Klasse6_AfterUpdate()
    'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse6 = "out" Then
    If Me!Klasse1 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse2 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse3 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse4 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse5 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
```

```
    End If
    If Me!Klasse7 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse8 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse9 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse6 = "nicht relevant"
        Exit Sub
    End If
End If
End Sub
```

```
Private Sub Klasse7_AfterUpdate()
    'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse7 = "out" Then
    If Me!Klasse1 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse2 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse3 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse4 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse5 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse6 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
        Exit Sub
    End If
    If Me!Klasse8 = "out" Then
        MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
        Me!Klasse7 = "nicht relevant"
```

```
      Exit Sub
   End If
   If Me!Klasse9 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse7 = "nicht relevant"
      Exit Sub
   End If
End If
End If
End Sub

Private Sub Klasse8_AfterUpdate()
   'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde

If Me!Klasse8 = "out" Then
   If Me!Klasse1 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse2 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse3 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse4 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse5 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse6 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse7 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse9 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse8 = "nicht relevant"
      Exit Sub
   End If
End If
End Sub

Private Sub Klasse9_AfterUpdate()
   'Fehlerroutine um sicherzustellen, dass nur eine Ausgabevariable deklariert wurde
```

```
If Me!Klasse9 = "out" Then
   If Me!Klasse1 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse2 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse3 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse4 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse5 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse6 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse7 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
   If Me!Klasse8 = "out" Then
      MsgBox "Sie haben eine Variable bereits als Ausgabe deklariert."
      Me!Klasse9 = "nicht relevant"
      Exit Sub
   End If
End If
End Sub


Public Function blnSumme(spalte As String, rs As Recordset) As Boolean
   'Ermittlung der nichtleeren Spalten der temporären Tabelle für den Datenexport
   '
   'spalte            = zu untersuchende Spalte
   'rs      = zu übergebendes Recordset der betroffenen Tabelle
   'Rückgabewert ist True wenn Spalte Werte enthält, sonst False

   Dim s As Double                                    'Summenwert der Spaltendaten

   rs.MoveFirst
   Do Until rs.EOF
      s = rs(spalte) + s
      rs.MoveNext
      If s <> 0 Then                                  'Wenn Summe>0 dann Spalte
relevant, True
```

```
            blnSumme = True
            Exit Function
        End If
    Loop
    blnSumme = False
End Function
```

14.5 Modul SNNS Auswertung

```
Option Compare Database
Option Explicit

Private Sub Befehl36_Click()
   'Schalter zum Speichern der Auswertungen in der Tabelle Auswertungen

Dim db As Database
Dim aus As Recordset

Set db = CurrentDb()
Set aus = db.OpenRecordset("Auswertung")

If MsgBox("Möchten Sie das Ergebnis speichern ?", vbYesNo) = vbYes Then
   aus.AddNew
   aus!Titel = InputBox("Bitte geben Sie der Auswertung einen Namen", "Speicherung", "Auswertung
SNNS")
   aus!Varianz1 = Me.Varianz1
   aus!Varianz2 = Me.Varianz2
   aus!Varianz3 = Me.Varianz3
   aus!stdabw1 = Me.std1
   aus!stdabw2 = Me.std2
   aus!stdabw3 = Me.std3
   aus!fehler1 = Me.fehler1
   aus!fehler2 = Me.fehler2
   aus!fehler3 = Me.fehler3
   aus!datum = Now()
   aus.Update
End If
aus.Close
DoCmd.Close A_FORM, "Varianz"
End Sub


Private Sub Suche_AfterUpdate()
   'Füllen der Datebfelder nach Auswahl eines Schlüssels aus der Listbox

Dim db As Database
Dim snns As Recordset

Set db = CurrentDb()
Set snns = db.OpenRecordset("Select Auswertung.* from Auswertung where Auswertung.SID =" &
Me!Suche)

Me!Varianz1.ControlSource = ""
Me!Varianz1 = snns!Varianz1
Me!Varianz2.ControlSource = ""
Me!Varianz2 = snns!Varianz2
Me!Varianz3.ControlSource = ""
Me!Varianz3 = snns!Varianz3

snns.Close
End Sub
```

14.6 SNNS Auswertung suchen

```
Option Compare Database
Option Explicit

Private Sub Befehl36_Click()
   'Speicherung der Auswertungsdaten in der Tabelle Auswertung

Dim db As Database
Dim aus As Recordset

Set db = CurrentDb()
Set aus = db.OpenRecordset("Auswertung")

If MsgBox("Möchten Sie das Ergebnis speichern ?", vbYesNo) = vbYes Then
   aus.AddNew
   aus!Titel = InputBox("Bitte geben Sie der Auswertung einen Namen", "Speicherung", "Auswertung
SNNS")
   aus!Varianz1 = Me.Varianz1
   aus!Varianz2 = Me.Varianz2
   aus!Varianz3 = Me.Varianz3
   aus!stdabw1 = Me.std1
   aus!stdabw2 = Me.std2
   aus!stdabw3 = Me.std3
   aus!fehler1 = Me.fehler1
   aus!fehler2 = Me.fehler2
   aus!fehler3 = Me.fehler3
   aus!datum = Now()
   aus.Update
End If
aus.Close
End Sub


Private Sub Suche_AfterUpdate()
   'Nach Auswahl aus der Listbox werden die Datenfelder mit den aktuellen Werten gefüllt

Dim db As Database
Dim snns As Recordset

Set db = CurrentDb()
Set snns = db.OpenRecordset("Select Auswertung.* from Auswertung where Auswertung.SID =" &
Me!Suche)

Me!Varianz1 = snns!Varianz1
Me!Varianz2 = snns!Varianz2
Me!Varianz3 = snns!Varianz3
Me.std1 = snns!stdabw1
Me.std2 = snns!stdabw2
Me.std3 = snns!stdabw3
Me.fehler1 = snns!fehler1
Me.fehler2 = snns!fehler2
Me.fehler3 = snns!fehler3
snns.Close

End Sub
```

Literaturverzeichnis

[1] Hans-Jochen Bartsch, *Taschenbuch Mathematischer Formeln*, Verlag Harri Deutsch Thun und Frankfurt/Main 1990

[2] Hans-Heinrich Bote, *Neuro-Fuzzy-Methoden*, Springer Verlag Berlin, Heidelberg 1998

[3] Patrick Hamilton, *Künstliche neuronale Netze, Grundprinzipien, Hintergründe, Anwendungen*, vde Verlag Berlin, Offenbach 1993

[4] Johannes Jörg, Horst Hielscher, *Evozierte Potentiale in der Klinik und Praxis*, Springer Verlag Berlin, Heidelberg 1997, 4. Auflage

[5] Christoph Klawun, *Turbo Pascal 5.5 vom Aufsteiger zum Insider*, Band 2, Addison Wesley Publishing Company Bonn, München 1990

[6] Monika Köhle, *Neurale Netze*, Springer Verlag Wien, New York 1990

[7] Klaus Peter Kratzer, *Neuronale Netze, Grundlagen und Anwendungen*, Carl Hanser Verlag München 1991, 2. Auflage

[8] Jeanette Lawrence, *Neuronale Netze, Computersimulation biologischer Intelligenz*, Systeme Verlag GmbH München 1992

[9] Burkhard Lenze, *Einführung in die Mathematik neuronaler Netze*, Logos Verlag Berlin 1997

[10] Detlef Nauck, Frank Klawonn, Rudolf Kruse, *Neuronale Netze und Fuzzy-Systeme*, Vieweg Verlag Braunschweig, Wiesbaden 1994

[11] Dan Patterson, *Künstliche neuronale Netze, Das Lehrbuch*, Prentice Hall Verlag GmbH München-Haar 1997

[12] Valluru B. Rao, Hayagriva V. Rao, *C++ Neural Networks and Fuzzy Logic*, MIS Press New York 1993

[13] Helge Ritters, Thomas Martinetz, Klaus Schulten, *Neuronale Netze*, Addison Wesley Publishing Company 1994

[14] Raul Rojas, *Theorie der neuronalen Netze*, Springer Verlag Berlin, Heidelberg, New York 1996

[15] Andreas Zell, *Simulation neuronaler Netze*, Addison Wesley Company Publishing Bonn 1994

[16] *Lehr- und Übungsbuch Mathematik, Band IV*, Verlag Harri Deutsch Thun und Frankfurt/Main 1990

Abbildungsverzeichnis:

Tabellenverzeichnis:

Erklärung

Gemäß §20 Absatz 8 der Prüfungsordnung des Fachbereiches Angewandte Informatik und Mathematik der Fachhochschule Fulda vom 10. Mai 1989 versichern wir, dass wir die vorliegende Diplomarbeit selbstständig angefertigt und keine anderen als die genannten Quellen und Hilfsmittel verwendet haben.

Die Diplomarbeit wurde gemäß der bereits erwähnten Prüfungsordnung als gemeinsame Arbeit von Gerhard Röhrig und Torsten Schreiber erstellt.
Die schriftliche Ausarbeitung der Kapitel 2, 3, 4, 7, 10 wurde von Gerhard Röhrig, die Kapitel 5, 6, 8, 9, 11, 12 von Torsten Schreiber zusammengestellt.
Die verbeibenden Kapitel 1 und 13 und das entwickelte Programm KNN 1.0 (Kapitel 14) wurden im Team angefertigt.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und wurde auch nicht veröffentlicht.

Mit der hochschulöffentlichen Auslage der Arbeit erklären wir uns einverstanden.

Fulda, den 24. Dezember 1998

Gerhard Röhrig Torsten Schreiber