Realisierung einer automatischen Klassifizierung von Körpertypen aus Messdaten einer automatischen 3D-Messkabine für die industrielle Maßkonfektion von Oberbekleidung

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Informatiker

an der
Fachhochschule für Technik und Wirtschaft Berlin

Fachbereich Wirtschaftswissenschaften II
Studiengang Angewandte Informatik

1. Betreuer:	Prof. Dr. Thomas Jung
2. Betreuer:	Dipl.-Ing. Lothar Paul

Eingereicht von Michael Stenschke

Berlin, 9. November 2004

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Motivation

In fast allen Bereichen unseres täglichen Lebens verstärkt sich heutzutage der Wunsch des Einzelnen nach mehr Individualität. Beim Autokauf kommt es nach der Marke und dem Modell auf charakteristische Ausstattungsmerkmale, wie Farbe, Motor, Art der Schaltung, Garnitur, Radio, Klimaanlage und diverse andere Extras, an. Selbst bei Fahrrädern gibt es inzwischen unzählige verschiedene Typen: Citybike, Mountainbike, Trekkingbike usw.

Besonders der Bekleidungsmarkt (einer der größten Konsumgütermärkte) befindet sich im Wandel zur individuellen Maßbekleidung mit vielfältigen persönlichen Gestaltungsmöglichkeiten. Um eine breite Kundenschicht für individuell hergestellte Produkte anzusprechen, müssen diese ein mit Massenprodukten vergleichbares Preisniveau haben und dem Kunden zeitnah zur Verfügung stehen.

Um den weltweiten Trend zur Maßkonfektion erfolgreich umzusetzen, ist es notwendig, die Prozesskette der Bekleidungsproduktion von der Körpervermessung bis zur Produktfertigung neu zu gestalten und an bestimmten Stellen zu automatisieren.

Körper- und Haltungsabweichungen einer einzelnen Person, wie z.B. Hängeschultern, Rundrücken, Hohlkreuz usw., von der Normalfigur sind bei einer automatischen Schnittkonstruktion nicht ohne weiteres umzusetzen, da sie zwar sichtbar, aber als Messdaten von Hand nur schwer zu erfassen sind. Daher mussten im Falle einer Abweichung die Schnittkonstruktionen bisher manuell entsprechend angepasst werden.

Ein weiteres Problem besteht bei der computergestützten Vorschau eines textilen Produkts anhand eines 3D-Modells, dessen Darstellung bei Haltungsabweichungen nur mit einer separaten Anpassung möglich war.

Künftig muss ein Weg gefunden werden, wie im Vorfeld ermittelt werden kann, welcher individuelle Körpertyp vorliegt. Diese Modelltypen müssen zunächst identifiziert werden. Anschließend muss ein System diese Typen unterscheiden können, um eine Zuordnung eines Körpers zu einem bestimmten Modell vornehmen zu können.

1.2 Betriebliches Umfeld

Die vorliegende Arbeit wurde im Rahmen aktueller Entwicklungsprojekte in der *Gesellschaft zur Förderung Angewandter Informatik* (GFaI) angefertigt. Die GFaI ist eine gemeinnützige Forschungsvereinigung im Bereich der angewandten Informatik. Sie fördert Forschung, Entwicklung und Training im Technologiebereich kleiner und mittlerer industrieller Unternehmen.

Darüber hinaus arbeiten mehr als 80 Ingenieure und Wissenschaftler in der GFaI selbst an Forschungs- und Entwicklungsprojekten.

Die Abteilung *3D-Datenverarbeitung* (3DDV) der GFaI beschäftigt sich mit berührungsloser 3D-Datenerfassung, Modellierung und Auswertung in Qualitäts- und Prozesskontrolle, digitaler Dokumentation und Präsentation dreidimensionaler Sachverhalte.

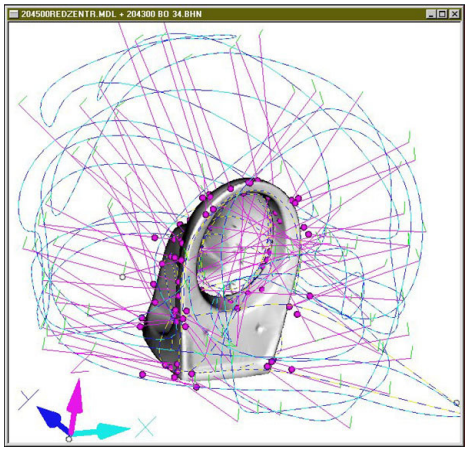

Abbildung 1: Robotersteuerung

Nach ersten Anwendungen im Maschinenbau, in der Architektur und der Robotik werden 3D-Modelle und -Darstellungen heute auch in der Medizin, der Textilbranche oder in Museen, Kunst und Kultur verwendet und benötigt.

Freigeformte räumliche Objekte werden im Bereich der 3DDV berührungslos vermessen. Durch komplexe mathematische Beschreibungen (Modelle) realer Objekte können solche Objekte am Rechner wirklichkeitsnah dargestellt, manipuliert, dokumentiert oder erneut vermessen werden. Mit Hilfe modernster *Rapid Prototyping* Technologien können physische Kopien der virtuellen Objekte erzeugt werden.

Aktuelle Projekte der Abteilung sind Steuerungsprogramme für Lackierroboter (Abbildung 1), Vermessen und Kontruieren von Zahnprothesen und das Vermessen von Personen für maßgeschneiderte Textilien.

Im Bereich 3DDV der GFaI ordnet sich die vorliegende Diplomarbeit in das Projekt „*Körpermaßerfassung unter dem Aspekt der Realisierung von automatischen Messkabinen*“ ein. Dieses ist nur ein Teilprojekt einer Forschungskooperation verschiedener Firmen, mit dem Ziel, individuelle Bekleidungserzeugnisse mit industriellen Verfahren herzustellen. Es soll der Aufbau einer weitgehend automatisierten „Textilen Kette“ erfolgen: von der automatischen Erfassung der Körpermaße, zur automatischen 2D-Schnitt-Konstruktion bis zum automatischen Zuschnitt und der Fertigung der Bekleidung.

1.3 Zielsetzung

Ziel der vorliegenden Diplomarbeit war es, eine geeignete Methode zu entwickeln, die aus über 30 gemessenen Körpermaßen verschiedene Körpertypen automatisch klassifiziert. Dazu muss zuerst definiert werden, welche Körpertypen klassifiziert werden sollen.

Die Körpertypen sind keinem Standard unterworfen, sondern jeder Schneider hat seine eigenen Definitionen von Typen. Dazu muss dieser bei jedem Kleidungsstück entscheiden, wie die Körpertypen Einfluss auf die Konstruktion nehmen. Bei einfachen Konstruktionen reichen vielleicht zwei unterschiedliche Typen, bei anderen sind es vier oder fünf.

Die parametrische Anpassung konstruierter Bekleidungsschnitte an die individuell erfassten Körpermaße unterliegt Grenzen. Daher müssen bereits die Schneider bei der Konstruktion von Bekleidungsstücken verschiedene Körpertypen unterscheiden. Diese Typen unterscheiden sich in Körperform und -haltung. Für den Schneider bedeutet dies zum Beispiel, nicht nur den Schnittlängen zu skalieren, sondern auch die Konstruktion zu ändern.

Für die zu lösende Klassifizierungsaufgabe werden unterschiedliche Ansätze in Theorie und Praxis miteinander verglichen.

Es war ein algorithmisches Konzept zur Klassifizierung von Körpertypen nach Anwendervorgabe zu entwickeln, dass heißt, bei wechselnden Schneidern oder Designern sollte sich das System den jeweils gewünschten Körpergrundtypen anpassen lassen.

Die Umsetzung des Konzeptes wird eine eigene Softwareentwicklung sein, die als Bibliothek in der Abteilung 3DDV der GFaI weiter benutzt werden kann.

1.4 Struktur der Arbeit

In Bezug auf die gesetzte Zielstellung wird in Kapitel 2 „Die Textile Kette“ zunächst der Ausgangspunkt beschrieben.. Es wird der Begriff der kundenindividuellen Massenproduktion (Mass Customization) erläutert. Über die Struktur des Prozesses „Die Textile Kette“ wird ein Überblick gegeben. Dabei wird noch mal besonders auf die bereits praktisch vorliegenden Komponenten „automatische Körpermaßerfassung“ und „parametrische Schnittkonstruktion“ eingegangen.

Der Hauptanteil der Diplomarbeit umfasst die Klassifizierungsproblematik in Kapitel 3. Es werden die für eine Klassifizierung notwendigen Grundlagen ermittelt, die für die folgenden, getesteten Klassifizierungen gültig sind. Drei verschiedene Wege der Klassifizierung werden vorgestellt und miteinander verglichen.

Das Kapitel 4 „Algorithmisch-programmtechnischer Teil“ beschreibt die Entwicklung und Implementierung von drei Prototypen, die jeweils eine Klassifizierungstechnik beherrschen. Die Prototypen werden mit verschiedenen Daten getestet, um so die beste Klassifizierung zu ermitteln. Die daraus folgenden Ergebnisse werden vorgestellt.

Die Zusammenfassung in Kapitel 5 bewertet den entwickelten Prototypen, zeigt dessen Stärken und Schwächen, und es werden Zukunftsperspektiven genannt.

2 Die Textile Kette

2.1 Mass Customization

Die Bekleidungsbranche hat sich in den letzten Jahren nicht nur strukturell verändert, sondern ist aufgrund der Kurzlebigkeit des Produktes „Mode“ einem ständigen Wandel unterworfen. In [Ganter] wird die derzeitige Situation sehr ausführlich beschrieben. Globale Beschaffungs- und Vertriebsstrukturen, Verlagerung von Produktionen ins Ausland, Verschärfung der Konkurrenzsituation auf den heimischen Märkten sowie ein kaum noch vorhersehbares Verbraucherverhalten haben die gesamte Bekleidungsbranche stark unter Druck gesetzt. So wurde nach Zukunftsperspektiven gesucht, die Bekleidungsprodukte in Deutschland wirtschaftlich herzustellen und erfolgreich zu vertreiben. Die industrielle Maßkonfektion stellt eine Erfolg versprechende Strategie für die Branche dar.

Wer braucht Maßkonfektion, also individuell angepasste Kleidung? Der Industriegüterbereich ist schon immer durch eine ausgeprägt individualisierte Produktion gekennzeichnet. Firmen stellen für andere Firmen sehr häufig Produkte nach deren Wünschen her. Im privaten Güterbereich ist eine zunehmende Individualisierung auf einen weit reichenden Wertewandel zurückzuführen. Mit steigendem Wohlstand, der sich u.a. in einem höheren Einkommen, mehr Freizeit und einem höheren Bildungsniveau manifestiert, steigt der Wunsch nach individuellen Produkten.

Diesen Zusammenhang beschreibt die Bedürfnispyramide (Abbildung 2 [Seid01, S. 18]). Von der Massenproduktion gesättigt, verlangte es den Menschen auch bewusst nach Qualität. Der nächste Schritt war, aus einem Produkt mehrere zu machen, in dem man verschiedene Varianten dieses Erzeugnisses anfertigte und anbot. Aber im Grunde hatten immer noch tausende von anderen Menschen das gleiche Produkt. Auch Wissenschaftler wie Beck [Beck86] oder Scitovsky [Scit89] halten die Massenproduktion schon seit langem für eintönig und neuen Ansprüchen nicht mehr angemessen, da *„das menschliche Bedürfnis nach Abwechslung und Neuheit genauso groß ist wie der Wunsch zu überleben. Die Massenproduktion hat ihren Reiz verloren, weil immer mehr Menschen die gleichen oder ähnliche Gegenstände besitzen“* [Four94].

Gleichförmige Massengüter, die bis heute den Prototyp der Produktion vieler Branchen bilden, sind deshalb immer weniger geeignet, die Wünsche und Bedürfnisse der Kunden zu befriedigen. Eine Individualisierung der Leistungserstellung bietet hier wesentliche Potenziale, wenn die Kosten das gleiche Niveau haben wie die Standardmassenprodukte.

Ein aktuelles Beispiel sind moderne Mobiltelefone. Sie sind Produkte, die es in sehr vielen Varianten gibt. Aber durch die veränderbaren Menüs und Klingeltöne kann sich jeder seine individuellen Vorlieben einstellen.

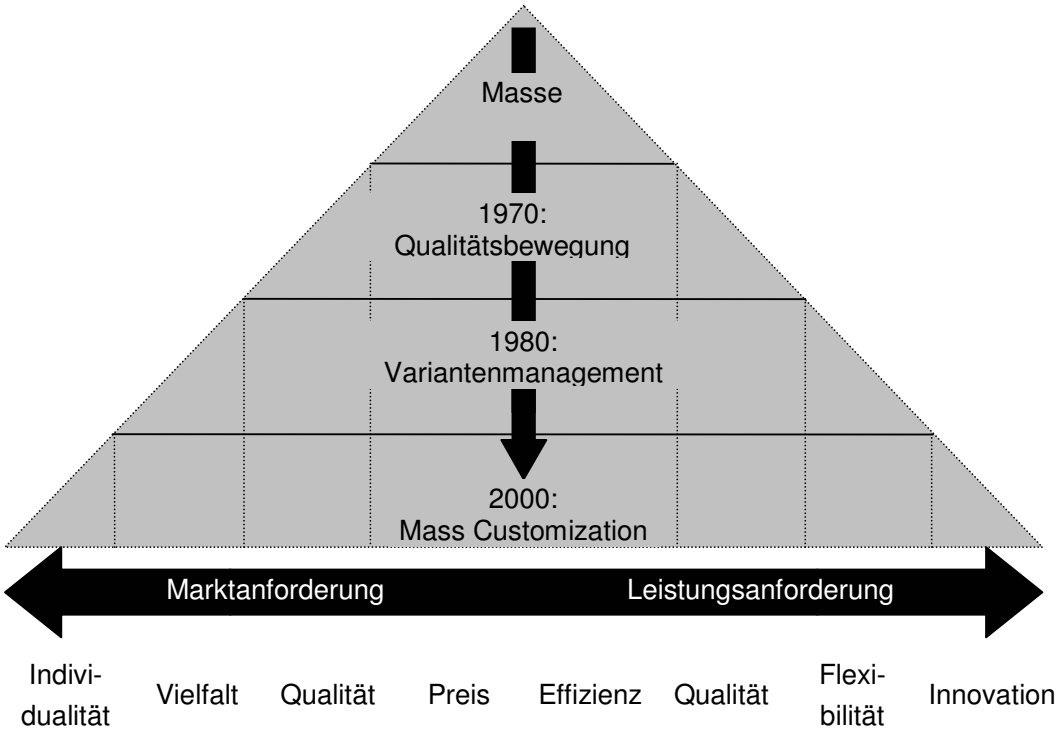

Abbildung 2: Bedürfnispyramide

Das Ziel der *Mass Customization*[1] ist die Produktion von Gütern und Leistungen für einen relativ großen Absatzmarkt, welche die unterschiedlichen Bedürfnisse jedes einzelnen Kunden zu diesen Produkten treffen. Dabei sollen die Kosten denen einer vergleichbaren massenhaften Standardherstellung entsprechen.

Der Startschuss für den Durchbruch von Mass Customization ist erst durch die weltweite Verbreitung des Internet möglich geworden. Die kundenindividuellen Informationen können durch das Netz schnell vom Kundenshop zum Produzenten des Produktes gesendet werden [Pill00]. Dieser kann mit einer rechnergestützten Fertigungstechnologie die Ware, optimal auf individuelle Wünsche angepasst, herstellen.

> *„Die individuelle Beziehung zu jedem einzelnen Kunden bietet dabei völlig neue Wege zur Steigerung der Kundenbindung im Rahmen eines intelligenten Customer Relationship Management (CRM). Denn ein nachhaltig erfolgreiches CRM-Konzept individualisiert nicht nur die Kommunikation, sondern ist auf die Wünsche und Ansprüche jedes einzelnen Kunden ausgerichtet."* [Pill00]

In den Medien wird auch schon über industriell gefertigte textile Maßkonfektion berichtet, aber in der preislichen Mittelklasse und bei den meisten Produzenten und Kaufhäusern ist diese noch nicht angekommen. Eine aktuelle Umfrage in der [Textil04] ergab, dass nur „*20% der Einzelhändler (TW-Testclub) konfektionierte Maßbekleidung*

[1] dt.: kundenindividuelle Massenproduktion

anbieten – vorrangig um sich von den Mitbewerbern abzugrenzen (79%) und um den Kunden einen besonderen Service (75%) bieten zu können“ (siehe Abbildung 3 [Textil04]). Darüber hinaus ist zu bemerken, dass auch beim überwiegenden Teil der bisher kommerziell verfügbaren Angebote mit individueller Körpermaßerfassung nicht wirklich maßkonstruierte Individualmodelle von Bekleidung erstellt, sondern lediglich vorhandene Basisgrößen nachträglich (teilweise manuell) variiert bzw. individuell angepasst werden.

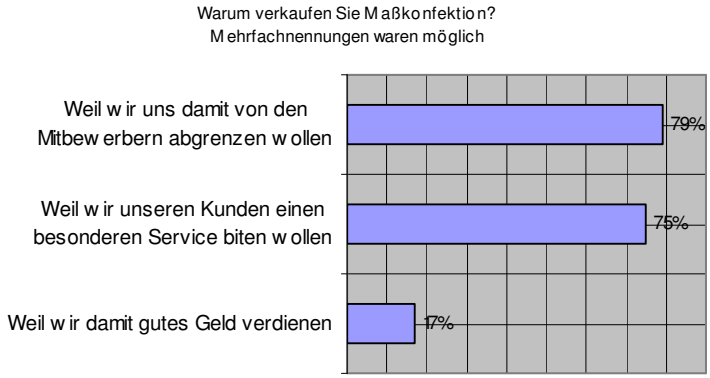

Abbildung 3: Maß kontra Masse

Eine kundenindividuelle Produktion kennt keine überschüssige Produktion durch Modellwechsel oder Modetrends.

Abbildung 4: Die Textile Kette

Die Textile Kette (Abbildung 4, GFaI-Flyer) beschreibt im Bereich der Mass Customization den Prozess der Herstellung von kundenindividuell angepasster Kleidung. Dazu gehören die Maßerfassung, die immer gleich und sehr genau erfolgen muss, die individuelle Schnitterstellung mit Körpertypwahl und Anpassung der Schnittparameter, die Fertigung und der Versand zum Kunden. In den folgenden Abschnitten wird auf die

Fertigung und der Versand zum Kunden. In den folgenden Abschnitten wird auf die Besonderheiten der Maßerfassung und Schnittanpassung eingegangen.

2.2 Lösungen für die berührungslose Messung der individuellen Körperform

Bereits in den 60er Jahren wurden von Kunden Fotoaufnahmen vor einem gerasterten Hintergrund erstellt. Die Schneider konnten im Nachhinein Informationen über den Körpertyp aus den Bildern beziehen und den Schnitt entsprechend anpassen. Es fehlte aber eine computergestützte Auswertung, die die Daten objektiver auswerten konnte. Erst mit steigender Rechnerleistung in den 80er und 90er Jahren wurden Systeme entwickelt, die den menschlichen Körper automatisch vermessen.

Man teilt die automatisierte Vermessung des Kunden in zwei Schritte:

- Optische, berührungslose Erfassung des Kunden (entweder über 2D-Body-Scanner oder 3D-Body-Scanner)
- Ermittlung der Körpermaße auf Grundlage der erfassten Messdaten (Konturbilder, Punktwolken)

Durch die Entwicklung von immer schnelleren und kosteneffizienteren Sensoriken und Rechnern können die heutigen automatischen Body-Scanner wesentlich schneller und detaillierter Informationen über die Körperform liefern, als dies mit dem Maßband möglich ist. Als Beispiel kann hierbei die Möglichkeit des Messens von Winkeln und Durchmessern genannt werden.

Manuelles Vermessen ist mit einem hohen Anteil systematischer und stochastischer Fehler behaftet. Während systematische Fehler meist auf falschen oder missverstandenen bzw. auf einer Vielzahl unterschiedlicher subjektiver Messvorschriften beruhen (Schulung des Personals), ist die Ablesegenauigkeit bei gebräuchlichen Messinstrumenten beschränkt. Dies wird unter anderem auch durch eine Weichteildeformation beim Vermessen begünstigt.

Der entscheidende Vorteil der berührungslosen Körpermaßerfassung ist die Reproduzierbarkeit als Grundlage der Stabilität der gesamten Prozesskette. Beobachtungen während der Projektarbeit an der GFaI-Messkabine haben gezeigt, dass sowohl bei der Vermessung der gleichen Person durch mehrere Fachleute als auch bei Vermessungsreihen durch die gleiche Person Toleranzen im Bereich zu mehreren Zentimetern auftreten.

Um die Körpermaße und -form der Personen automatisch zu erfassen, wurden zwei Ansätze verfolgt: *2D-Body-Scanner* und *3D-Body-Scanner*.

Beim 2D-Body-Scanner werden mindestens zwei Ansichten des Kunden aufgenommen. Eine Ansicht von vorne und eine von der Seite. Aus den gewonnenen Konturen werden Längen-, Breiten- und Tiefenmaße ermittelt. Die Umfangsmaße werden über statistische Relationen und Menschmodellwissen berechnet.

Die Firma *Puls* verwendet ein aufwendiges Spiegelsystem, so dass eine simultane Aufnahme der Vorder- und Seitenansicht der Messperson erreicht wird. Außerdem musste der Kunde mit Leuchtstangen in Achseln und im Schrittbereich ausgeleuchtet werden [Seid01].

Ein vollständigeres und meist genaueres Abbild des Kunden liefern 3D-Body-Scanner. Die bekanntesten nutzen das Laser-Lichtschnittverfahren mit optischer *Triangulation*. Dazu werden um den Körper des Kunden verteilt Sensorköpfe vertikal verschoben. Die aufgenommenen 3D-Punkte werden im Computer ausgewertet und zu einem konsistenten Gesamtbild verschmolzen.

Die französische Firma *TELMAT* verwendete früher auch einen 2D-Scan mit gleichzeitiger Front- und Seitenansichtaufnahme [Seid01]. Inzwischen verwendet ihr System SYMCAD aber ein 3D-Scan-Verfahren, welches die Person in einem „Augenblick“ [SYMCAD] aufnimmt.

Der *WB4 – Whole Body Scanner* der amerikanischen Firma *Cyberware* und das System *VITUS* von Human Solutions GmbH [Human] sind in der Lage, annähernd vollständige Daten für die Körpergeometrie zu liefern. Beim System VITUS wird der Kunde innerhalb von 20 Sekunden millimetergenau mit Lasern abgetastet. Dabei fallen mehrere Millionen 3D-Punkte an. Im Vergleich zu 2D-Verfahren verursacht vor allem die zeitsynchrone, vollständige Erfassung einen hohen technischen Aufwand, der sich in hohen Systemkosten niederschlägt (Abbildung 5 [Human]).

Abbildung 5: System VITUS von Human Solutions

Mit Infrarot-Licht arbeitet der *BL-Scanner* der Firma *Hamamatsu* für die spezielle Aufgabenstellung der *Anthropometrie*-Erfassung.

Bei einem 3D-Scan-Verfahren werden vom Menschen komplette 3D-Abbilder geschaffen. Das ist brauchbar für 3D-Avatare, ein Schneider aber braucht einen standardisierten Satz von Körpermaßen, um die jeweiligen Grundschnitte effektiver anzupassen.

Aus einer 3D-Figur, die aus sehr vielen Messpunkten besteht, müssen noch nach umfangreichen Algorithmen die wichtigen brauchbaren Daten berechnet werden.

2D-Body-Scanner sind zwar preiswert, besitzen aber nur eine geringe Genauigkeit und weisen wesentliche Einschränkungen hinsichtlich der messbaren Größen auf (fehlende Tiefeninformationen). 3D-Body-Scanner besitzen eine sehr hohe Genauigkeit, haben aber u.a. durch den Einsatz von augensicheren Lasern einen für den Einzelhandel zu hohen Preis.

Um die Vorteile, der beiden unterschiedlichen Ansätze, niedrige Kosten und hohe Genauigkeit zu kombinieren, verwendet die automatischen Messkabine der GFaI ein spezielles, automatisch ablaufendes, aktives 2D-Kontur- und 3D-Kurvenerfassungssystem. Mit dem Messsystem *BodyFit 3D* werden vollautomatisch über 30 vordefinierte Körpermaße innerhalb eines Messablaufes von etwa 1,5 min erfasst. Die Einzelmessungen dauern dabei nur Bruchteile von Sekunden. Der Großteil der 1,5 min wird für die akustischen und visuellen Anweisungen für den Kunden benötigt, um ihm die richtige Messposition zu verdeutlichen.

Zur Ermittlung der zur Maßkonfektion benötigten Körpermaße wurde ein zweistufiges Messverfahren entwickelt. Im ersten Schritt werden die Kontur erfasst, Körpergröße und markante Konturpunkte, wie Schritt oder Achseln erkannt und körpermaßrelevante Bereiche, z.B. Brust und Taille, selektiert. Dabei wird auch die richtige Position der Person überprüft und gegebenenfalls korrigiert [Heuw02].

Im zweiten Schritt werden Lichtstreifen an die eingegrenzten zu vermessenden Bereiche projiziert und die Abbildungen der Projektionsmuster ausgewertet. Bei der 3D-Erfassung kommt normales Weißlicht zum Einsatz, was möglichen gesundheitlichen Bedenken gegenüber Laserstrahlen die Grundlage entzieht, außerdem sind Systeme mit einem Lichtprojektor um einiges preiswerter als solche mit Laser, die auch mechanische Präzisionsachsen benötigen.

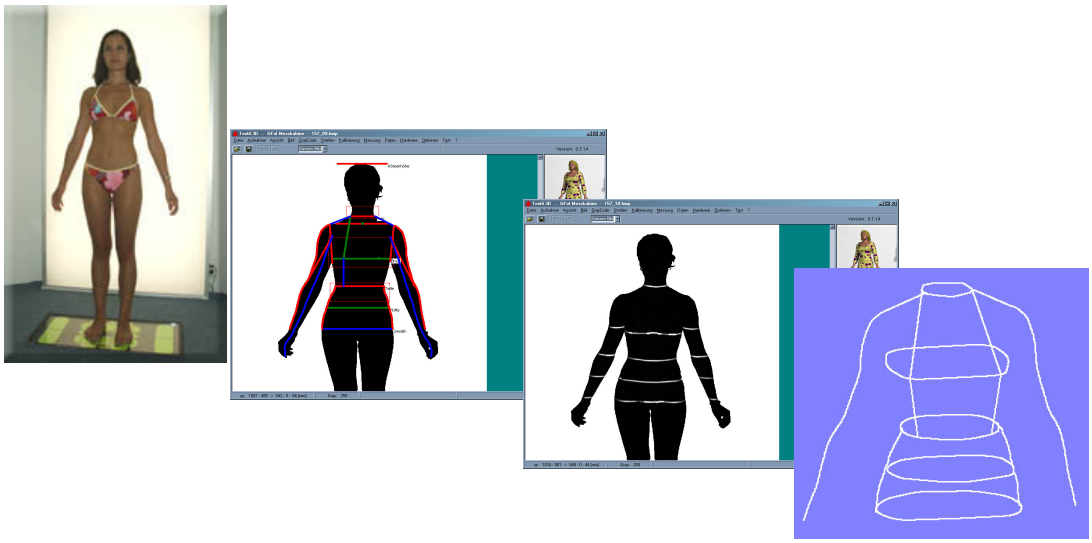

Abbildung 6: Messprozess von BodyFit 3D der GFaI

Mit dem Lichtschnittverfahren werden fast alle erforderlichen Konstruktionsmaße durch eine begrenzte Anzahl von Messpunkten (Polylines) direkt erfasst, ohne den gesamten Körper durch Millionen von Punkten vielfach redundant aufzunehmen und auszuwerten. Ergebnisse dieser Auswertemethodik sind dreidimensionale Objektkoordinaten und die von der Bekleidungskonstruktion geforderten Körperparameter, Kurven und Maße [BodyFit].

2.3 Parametrische Konstruktion

Für die Erstellung eines Modellschnitts für die Bekleidungsindustrie oder die Maßschneiderei ist immer ein Grundschnitt erforderlich. Dieser bildet die Basis für die Modellgestaltung, wobei bereits die erforderlichen Bewegungszugaben und Nähte berücksichtigt sind. Bei der parametrischen Konstruktion werden einzelne Maße und ihre Verhältnisse als variable Parameter definiert, mit denen der Schnitt individuell geändert bzw. an die konkreten Körpermaße angepasst werden kann.

Für die Schnitterstellung von Konfektionsgrößen werden heute in der Industrie nahezu flächendeckend computergestützte Schnittsysteme eingesetzt. Dabei steht neben der Konstruktion der Schnittteile die Erzeugung aller Größen eines Größensatzes durch Vergrößerung oder Verkleinerung der Schnitte einer Basisgröße (*Gradierung*) im Mittelpunkt. Die Parameter zum Ändern der Konstruktion sind die Maße, die man vom Kunden bekommt und mit einer Größentabelle abgleicht [Seid01].

Im Bereich der Maßkonfektion ist der Einsatz der computergestützten Konstruktion noch beschränkt. Heute ist vornehmlich die so genannte Maßgradierung im Einsatz. Dabei wird eine den Kundendaten am nächsten liegende Größe entsprechend verändert. Für komplexere Modifikationen, wie sie bei der Schnittanpassung z.B. für Rundrücken notwendig sind, stehen meist nur einfache Funktionen wie das Aufdrehen der Schnittteile zur Verfügung.

Um alle möglichen Maßabweichungen und Haltungsvarianten abzudecken, müssen unterschiedliche Ausprägungsstufen von körpermaß- und haltungsbedingten Schnittveränderungen realisiert werden. Doch trotz umfassender Grundlagen und der Bereitstellung ausreichender Gradiervarianten sind die Möglichkeiten der Maßgradierung beschränkt. Die Anpassung von Bekleidungsschnitten ist nur für Körperformen möglich, die nicht wesentlich von den gängigen bzw. der Konstruktion zugrunde gelegten Größenproportionen abweichen. Bei extremen Abweichungen der Körpermaße und -haltung vom „Normalen“ ist eine manuelle, qualitative Veränderung der Schnittteile erforderlich [Kirch01].

Für die Konstruktion eines Schnittes benötigt man genaue Maßangaben. Diese Maße werden am Körper gemessen und gleichzeitig auch zum Teil nach Formeln berechnet. Die Maße werden bei der herkömmlichen Handvermessung in drei Gruppen eingeteilt, die Hauptmaße, die am Körper gemessen werden müssen, Hilfsmaße, die man messen oder berechnen kann, und Hilfsmaße, die man errechnen soll. Zu den Hauptmaßen zählen Körperhöhe, Brustumfang, Taillenumfang und Hüftumfang. Ohne diese

Maße kann kein Schnitt erstellt werden. Mit der zweiten Gruppe von Maßen kann man Abweichungen am Körper feststellen, die man sonst übersehen hätte. Die dabei errechneten Maße bieten auch eine gute Kontrollmöglichkeit, da man sich auch leicht vermessen kann. Zu diesen Hilfsmaßen zählen unter anderem Rückenhöhe, Rückenbreite, Hüfttiefe, Rückenlänge und Vordere Länge. Die letzte Gruppe von Hilfsmaßen ist auch messbar, aber bei ihnen ist die Wahrscheinlichkeit einer möglichen Fehlmessung zu hoch, so dass diese errechnet werden sollen. Zu ihnen gehören die Halsspiegelbreite, der Armlochdurchmesser und die Brustbreite [Stieg92].

Für viele Größen gibt es in der Damen-Oberbekleidung (DOB) Größentabellen. Dabei werden normalhüftige, schmalhüftige und starkhüftige Größen unterschieden. Diese decken bereits einen Großteil der Größen ab, die benötigt werden.

Die Grundelemente automatischer Konstruktionssysteme verschiedener Hersteller sind nahezu identisch. Sie gliedern sich in die Teilelemente Körpermaße, Konstruktionsmaßableitung aus den Körpermaßen und Beschreibung des Konstruktionsablaufes.

2.4 Möglichkeiten und Grenzen der parametrischen Anpassung von Schnitten: Körpertypen

Schnittkonstruktionen können nach bestimmten Regeln und Normen geändert werden. Mit Hilfe des Maßsatzes errechnet man sich die fehlenden Hilfsmaße und zur Konstruktion kommen je nach Modell ein paar Zentimeter dazu oder weg (Zugaben), oder die Abnäher werden variiert.

Abweichungen vom Standardfall wird fast jede Kundin oder jeder Kunde haben, aber nicht jede dieser kleinen oder auch größeren Abweichungen muss unbedingt eine Abänderung des Grundschnittes erfordern. Eine Hängeschulter kann man z.B. auch mit einem Schulterpolster ausgleichen.

In dem Bestreben einen möglichst guten Sitz des Kleidungsstückes zu erreichen, werden manchmal zu viele Passformabänderungen vorgenommen. Dies führt unter Umständen dazu, dass das Kleidungsstück zwar gut passt, das aber gleichzeitig die Figur so ungünstig erscheint, wie sie auch tatsächlich ist. Die folgenden Skizzen sollen unterschiedliche Körperhaltung verdeutlichen.

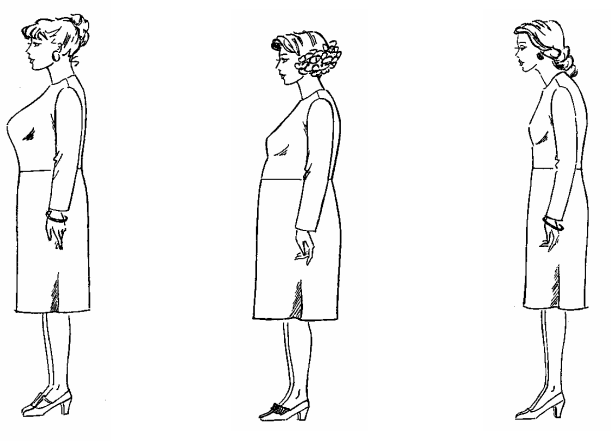

Abbildung 7: Körpertypen

In Abbildung 7 aus [Stieg92] sind verschiedene Körpertypen abgebildet, links sieht man eine Frau die eine so genannte starke Brust und ein flaches Gesäß hat. In der Mitte ist eine Leibfigurperson abgebildet und rechts eine Figur mit Rundrücken.

Mit den Schnitten, die für die normalen Konfektionsmaße generiert wurden, kann man nur einen Teil der Kunden zufrieden stellen, es gibt auch einen erheblichen Anteil von Figuren, die von der Norm abweichen. Bei ihnen sind z.B. die Längen- oder die Weitenmaße in keinem normalen[2] proportionalen Verhältnis zueinander. Für diese schwierigen Körpertypen ist bereits während des Prozesses der Konstruktion des Bekleidungsstückes eine entsprechend angepasste Schnittgenerierung wünschenswert.

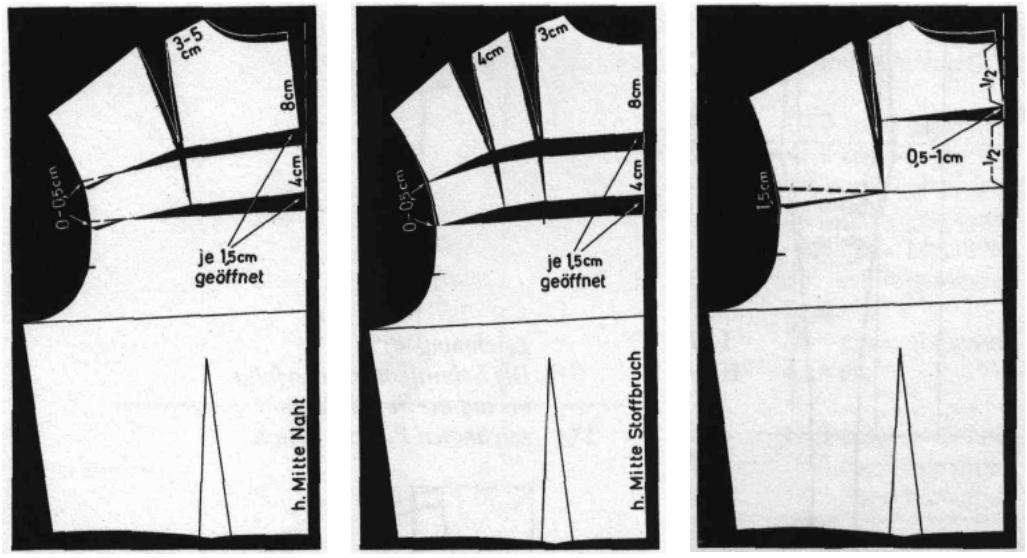

Abbildung 8: Schnittkonstruktionen

[2] normal – dem Durchschnitt entsprechend

In Abbildung 8 aus [Stieg92] sieht man drei unterschiedliche Konstruktionen einer Bluse für verschiedene Körpertypen. Auffällig ist die unterschiedliche Anzahl von Abnähern (Einschnitte).

Für einige Kleidungsstücke werden nur wenige Schnittänderungen benötigt, dort reichen 2-3 Körpertypen aus. Bei anderen Kleidungsstücken wird wesentlich differenzierter zwischen den Körpertypen unterschieden, es können 5 bis 8 Typen betrachtet werden [Stieg92].

3 Klassifizierung

3.1 Klassifizierung und Merkmalsräume

Als Klassifizierung oder Klassifikation bezeichnet man einen Vorgang zur Einteilung von Objekten aufgrund ihrer Eigenschaften in verschiedene Klassen oder Kategorien. Die klassifizierbaren Objekte können sowohl Gegenstände als auch Sachverhalte sein. Unterscheiden kann man diese Objekte auf ganz unterschiedliche Weise, z.B. nach biologischen, mathematischen oder auch statistischen Werten.

Der Mensch klassifiziert ständig, um seine Umwelt besser zu verstehen und rasch reagieren zu können. Die ersten Klassifizierungen, die ein Lebewesen durchführt sind: „Essbar/Nichtessbar“ und „Gefährlich/Ungefährlich“. Bei modernen Menschen ist die Klassifizierung alleine für „Essen“ inzwischen extrem komplex geworden (Tante Emma-Laden, Supermarkt, Onlinebestellung; Obst, Brot, Fleisch, Wurst, Milchprodukte …; Salami, Leberwurst, Teewurst, …; Normalsalami, Edelsalami, Pfeffersalami, Paprikasalami,…; ungeschnitten/geschnitten; Menge; genug Geld; Geschmack; usw.). Auch Hören und Lesen sind Beispiele für Klassifizierung. Dies alles macht einen Teil unserer Intelligenz aus.

Computer sollen den Menschen bei der Bewältigung seiner Aufgaben unterstützen, dazu gehört auch die Klassifizierung. Die Klassifizierung wird zum Gebiet der künstlichen Intelligenz (KI) zugeordnet. Zur künstlichen Intelligenz gibt es verschiedene Definitionen, die zu einem den Menschen als Vergleich heranziehen oder zum Anderen die einzelnen Arbeitsgebiete der KI aufzählen. In [Lämm01, S. 13] wird künstliche Intelligenz wie folgt definiert:

> *„Teilgebiet der Informatik, welches versucht, menschliche Vorgehensweisen der Problemlösung auf Computern nachzubilden, um auf diesem Wege neue oder effizientere Aufgabenlösungen zu erreichen.“*

Für die automatische Klassifikation wurden verschiedene Verfahren entwickelt. Für jedes Verfahren gibt es noch mal unterschiedliche Algorithmen. Die bekanntesten Klassifikationsverfahren sind:

- k-Nächste-Nachbarn-Klassifikation - kNN
- Entscheidungsbaum-Klassifikation - EB
- Bayes-Klassifikation
- künstliche Neuronale Netze - NN
- Support Vector Machines - SVM

Zu Beginn einer Klassifizierung muss eine endliche Menge C = {C_1,…, C_j} definiert werden. Jedes Element c_i bildet eine Klasse, in der Objekte eingeordnet werden können.

Die einzelnen, zu klassifizierenden Objekte können durch verschiedene Merkmale unterschieden und spezifiziert werden. Diese Merkmale müssen vorher ermittelt werden. In diesem Bereich wird „*Data Mining*“ betrieben.

Der Begriff „*Data Mining*“ (dt. Datenbergbau) im Zusammenhang mit Klassifizierung steht für den Prozess, große Mengen an Daten dahingehend zu analysieren, ob Strukturen und Beziehungen vorliegen. Das Ziel ist es eine Vorhersagbarkeit für neue Daten zu erreichen. Als Beispiel für große Datenmengen sind dabei Muster wie Bilder und Spracherkennung zu nennen.

Unterschieden wird in qualitative und quantitative Merkmale. Qualitativ meint dabei als Beispiel Farben wie rot, grün und blau oder auch Formen wie Quader und Kreis. Hingegen quantitativ heißt die Abzählbarkeit der Merkmale, z.B. der Parameter Taillenumfang ist abzählbar im Zentimeterbereich (55,5 cm, 60,0 cm). Um qualitative Merkmale mathematisch zu unterscheiden, können diese auch nach unterschiedlichen Modellen quantisiert werden (Farben im RGB- oder HSI-Farbmodell).

Alle zu klassifizierenden Objekte bilden die Menge O. Die Objekte werden in der Literatur als Muster bezeichnet, da sie durch ihre Merkmale ein eindeutiges Muster darstellen. Alle Merkmale x_n eines Objektes $o_i \in O$ werden in einem Vektor $\vec{x}$ gespeichert:

$$\text{Merkmalsvektor}: \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in M^n$$

Eine Gruppe von n Merkmalen bildet einen n-dimensionalen Raum M, der als Merkmalsraum bezeichnet wird. Für die Darstellung von Merkmalsräumen werden in dieser Arbeit zum leichteren Verständnis meist nur zweidimensionale Merkmalsräume verwendet.

Ein trainiertes Klassifizierungssystem nennt man Klassifikator, und K(o) ist die Klassifizierung des Objekts o durch den Klassifikator K.

Damit die Algorithmen schnell und der benötigte Speicherplatz gering bleibt, muss man unwichtige Daten aus dem Vektor löschen. Zum Beispiel benötigt man nach derzeitigem Wissen die Haarfarbe, die Augenfarbe, den Namen und die Adresse einer Person nicht, um dessen Körperhaltung zu ermitteln. Diese erste Analyse der Merkmale ist immer noch von Menschen durchzuführen.

Es müssen aber nicht alle scheinbar unwichtigen Merkmale entfernt werden. Die, die keinen Einfluss auf die Ergebnismenge haben, sollten von einem guten Klassifikator auch nicht beachtet werden. Es könnten auch Merkmale entdeckt werden, an die man vorher nicht gedacht hat. Es lassen sich somit vielleicht bisher unbekannte Zusammenhänge in den Daten identifizieren oder Regeln für die Entscheidungsunterstützung

aus den Daten generieren. Nach einer Analyse des erstellten Klassifikators kann der Merkmalsraum so optimiert werden.

Für einige Klassifizierungen, wie kNN und NN, müssen die Merkmale, die der Algorithmus für die Klassifikation berücksichtigen soll, normiert werden. Angenommen, ein Merkmal liegt im Bereich zwischen 100 cm und 220 cm (Körperhöhe) und ein anderes zwischen 1 cm und 20 cm (Abstand Hals zur Rückenlinie), dann könnte bei einem NN, das erste Merkmal dazu tendieren, das zweite zu dominieren, da sein Einfluss auf die Eingabe zu einer Einheit größer ist. Bei kNN dagegen würde das zweite Merkmal dominieren, weil hier die Distanzen zwischen zwei Objekten von Bedeutung sind. Mehr dazu wird in den jeweiligen Kapiteln der Algorithmen beschrieben.

Eine bewährte Methode ist es, die Daten so zu normieren, dass sie innerhalb eines bestimmten Bereichs liegen. *„Der Bereich zwischen 0,1 und 0,9 wird beispielsweise oft für die sigmoide Aktivierungsfunktion[3] verwendet, um zu verhindern, dass das Netz zum Stillstand kommt, weil es außerhalb seines Operationsbereichs arbeitet."* [Call03] Sigmoide Funktionen sind in ihrer grafischen Darstellung dem Buchstaben „S" sehr ähnlich.

Um ein Merkmal in den Bereich zwischen 0,1 und 0,9 zu skalieren, kann folgende Formel benutzt werden:

$$y = \frac{0{,}9 - 0{,}1}{x_{\max} - x_{\min}} x + \left(0{,}9 - \frac{0{,}9 - 0{,}1}{x_{\max} - x_{\min}} x_{\max} \right) \qquad (1)$$

Hierbei ist x der originale Wert des Merkmals, x_{min} ist der kleinste Wert den das Merkmal annehmen kann und x_{max} der größte Wert. Das Ergebnis y ist der neue skalierte Wert. Dadurch dass x_{min} auf 0,1 und x_{max} auf 0,9 gesetzt werden, sind nach unten und oben auch noch 10 Prozent Reserve, falls doch mal ein Merkmal über die gesetzten Grenzen tritt. Man kann die Bereichsgrenzen natürlich für andere Klassifizierungen auch anders skalieren.

Um einen Klassifikationsalgorithmus optimal einzusetzen, braucht man Trainingsdaten. Gegeben sei eine Menge TR von Trainingsdaten mit den Merkmalen $x_1,\ldots,x_n$ und zusätzlich dem Klassenattribut c, das jedes Trainingsobjekt einer der bereits bekannten Klassen $C_1,\ldots, C_j$ zuordnet. Diese Trainingsmenge TR muss eine repräsentative Stichprobe von Daten aller möglichen Klassen darstellen:

$$TR \subseteq O$$

Dabei sind besonders die Trainingsdaten interessant, die in den Grenzregionen von unterschiedlichen Klassen liegen. Durch diese Objekte kann die Genauigkeit der Klassifizierung enorm gesteigert werden.

[3] sigmoide Funktionen - siehe Glossar

Die Abbildung 9 zeigt dieses Problem. Auf dem linken Bild ist die Grenzregion total unterbesetzt. Die Zuordnung eines Objektes zu einer Klasse kann nicht eindeutig durchgeführt werden. Auf dem rechten Bild dagegen ist die Zuordnung kein Problem. Die Zuordnung von Objekten kann eindeutiger durchgeführt werden.

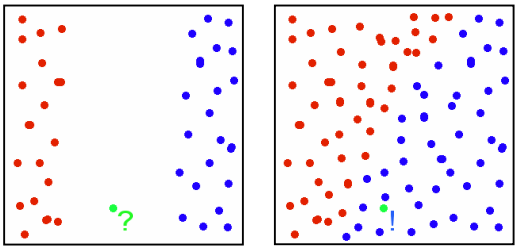

Abbildung 9: Grenzregionen

Bei den diversen Lernstrategien muss auch die Genauigkeit des Ergebnisses betrachtet werden. Es darf nicht zu genau auf die Trainingsdaten abgebildet werden. Das Klassifikationsmodell kann dann zu sehr für die Trainingsmenge TR optimiert sein und liefert auf die Gesamtheit der Daten aber eventuell schlechtere Ergebnisse. Dieses Problem bezeichnet man auch als „overfitting“.

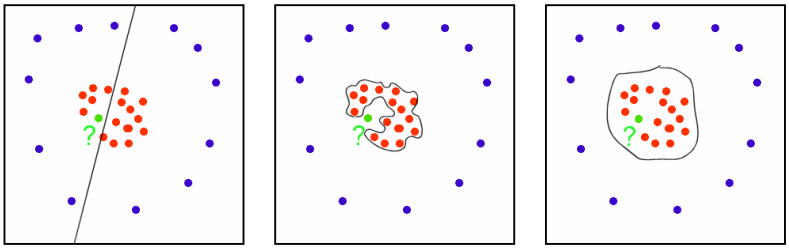

Abbildung 10: Generalisierung

Neben dem „overfitting“-Problem gibt es auch ein „underfitting“-Problem. Dabei wurde das System dann zu ungenau trainiert. Das Ziel ist es, das System so zu trainieren, dass neue Daten eindeutig klassifiziert werden können. Der Klassifikator muss fähig sein zu generalisieren. Mit Generalisierung wird die Fähigkeit eines Klassifikators bezeichnet, mit Daten korrekt zu arbeiten, die es während des Trainings nicht gesehen hat. In Abbildung 10 wird dies illustriert. Auf dem linken Bild handelt es sich um „underfitting“, das System ist zu ungenau trainiert. Im mittleren Bild wurde zu genau auf die Trainingsdaten trainiert, so dass neue Objekte schnell falsch klassifiziert werden können. Auf dem rechten Bild hingegen wurde so gut trainiert, dass auch neue unbekannte Objekte richtig klassifiziert werden können.

Der erfolgreiche Einsatz der automatischen Klassifikation hängt wesentlich von der Homogenität des Gebietes ab.

- Großflächige, reliefarme Gebiete (Landwirtschafts- und Waldgebiete) sind hervorragend geeignet.
- Inhomogene, komplexe, stark variierende Gebiete (Städte) bereiten größere Probleme.

Der größte Nachteil ist, dass es sich in jedem Fall um eine Zusammenfassung und Vereinfachung von komplexen Sachverhalten der Natur handelt. Daher gibt es keine hundertprozentig richtige Klassifizierung.

Die Güte des Klassifizierungsalgorithmus und der Merkmale lassen sich vor ihrem endgültigen Einsatz mit einer Testmenge TE überprüfen:

$$TE \subseteq O$$

Zu den Objekten von TE muss ebenso wie bei TR die Klassenzugehörigkeit bekannt sein. Die Testmenge sollte ähnlich aufgebaut sein wie die Trainingsmenge, also besonders die Grenzregionen überprüfen. Die Testmenge sollte aber nicht zur Konstruktion des Klassifikators verwendet worden sein, weil dass das Ergebnis verfälschen kann.

Um die Qualität des Klassifikators zu beurteilen, hilft es die Ergebnisse einer Klassifikation in eine Konfusionsmatrix einzutragen (Abbildung 11). Mit dieser Matrix lassen sich folgende Kennzahlen für die Güte der Klassifikation berechnen: Accuracy, Classification Error, Precision und Recall [Köst03].

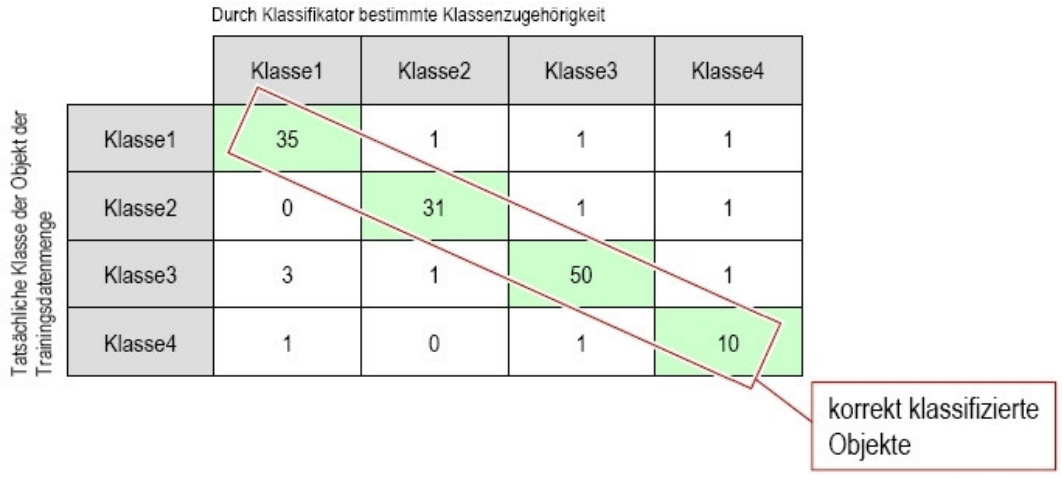

Abbildung 11: Ergebnisintegration in Konfusionsmatrix

Sei C(o) die tatsächliche Klasse des Objekts o, dann ist die Klassifikationsgenauigkeit (classification accuracy) von K auf TE:

$$G_{TE}(K) = \frac{\left|\left\{o \in TE \middle| K(o) = C(o)\right\}\right|}{\left|TE\right|} \tag{2}$$

Je höher G_{TE} ist, umso besser ist die Klassifikation So lässt sich dann auch der offensichtliche Klassifikationsfehler (appearent classification error) mit (3) ermitteln.

$$F_{TE}(K) = \frac{\left|\left\{o \in TE \middle| K(o) \neq C(o)\right\}\right|}{\left|TE\right|} \tag{3}$$

Beide Berechnungen beziehen sich allerdings nur auf die Testmenge TE, diese sollten dem tatsächlichen Klassifikationsfehler (true classification error) aber sehr nahe kommen. Der tatsächlichen Klassifikationsfehler wird bestimmt durch:

$$F_{O}(K) = \frac{\left|\left\{o \in O \middle| K(o) \neq C(o)\right\}\right|}{\left|O\right|} \tag{4}$$

Das Problem hierbei ist jedoch, dass es bei unbekannten Objekten keine Möglichkeit gibt, die wahre Klasse und damit potentielle Fehlklassifikationen zu erkennen. Aus der Statistik weiß man, dass sich die offensichtliche Fehlerrate der wahren Fehlerrate annähert, wenn man genügend Trainings- und Testobjekte zur Verfügung hat, so dass diese die Gesamtheit der Daten widerspiegeln [FKPT97]. In realen Problemstellungen ist es jedoch häufig so, dass die Menge der Trainingsdaten beschränkt ist. Es sind also Verfahren gesucht, mit denen man anhand einer beschränkten Menge von Trainingsobjekten eine Fehlerrate errechnen kann, die möglichst nah an der wahren liegt.

Eine sehr weit verbreitete Lösung ist die *n-fold Cross Validation*. Dabei wird als erstes die Trainingsmenge TRin n gleich große Teilmengen aufgeteilt. Dann werden n-1 Teilmengen zum Trainieren des Netzes verwendet und die übrig gebliebene Teilmenge zum Testen der Fehlerrate. Das Trainieren wird so oft wiederholt, bis jede Teilmenge als Trainingsmenge und Testmenge verwendet wurde. Die entstandenen Modelle und auch die einzelnen Klassifikationsfehler werden kombiniert, so dass man ein Gesamtmodell und einen Gesamtklassifikationsfehler erhält (Abbildung 12).

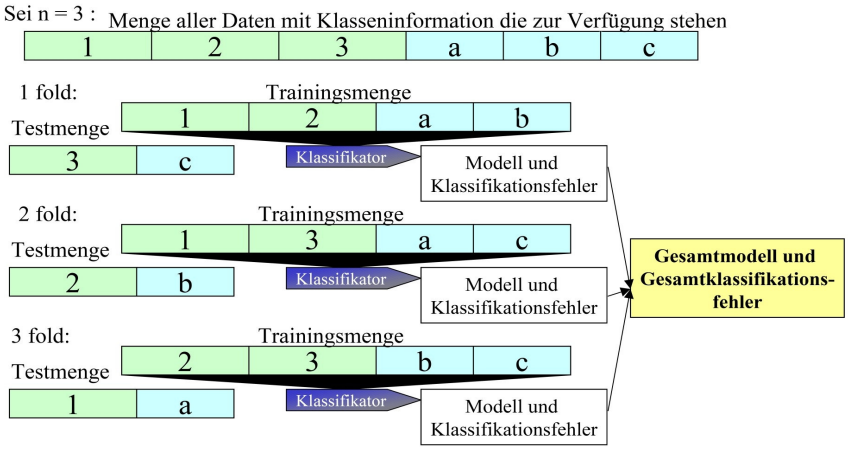

Abbildung 12: n-fold cross Validation

Weitere statistische Gütemaße für Klassifikatoren sind Precision und Recall. Precision beschreibt die Anzahl der Objekte einer Klasse, die richtig erkannt wurden, im Verhältnis zu allen Objekten dieser Klasse. Damit wird eine Zeile in der Konfusionsmatrix beschrieben. Sei $T_i = \{o \in TE | C(o) = i\}$, dann ist:

$$\text{Precision}_{TE}(K,i) = \frac{|\{o \in T_i | K(o) = C(o)\}|}{|T_i|} \tag{5}$$

Recall beschreibt die Anzahl der zu einer Klasse zugeordneten Objekte, die richtig erkannt wurden, im Verhältnis zu allen Objekten, die dieser Klassen zugeordnet wurden. Es wird hier eine Spalte in der Konfusionsmatrix näher untersucht. Mathematisch beschrieben: Sei $C_i = \{o \in TE | K(o) = i\}$, dann ist:

$$\text{Recall}_{TE}(K,i) = \frac{|\{o \in C_i | K(o) = C(o)\}|}{|C_i|} \tag{6}$$

Es gibt aber noch mehr Kriterien, die für oder gegen die Wahl eines Klassifizierungsalgorithmus sprechen:

- Kompaktheit des Modells
- Interpretierbarkeit des Modells
- Effizienz
- Skalierbarkeit für große Datenmengen
- Robustheit

Bei der Kompaktheit eines Modells geht es um die Größe dieses Modells, z.B. die Größe eines Entscheidungsbaums. Ein besonders großes Modell kann ein k-Nächste Nachbarn-Modell werden, es speichert alle Trainingsdaten.

Für die Wahl eines Algorithmus kann auch die Interpretierbarkeit von entscheidender Bedeutung sein. Wenn der Endbenutzer eine Klassifizierung nachvollziehen muss oder einmal ohne einen Computer klassifizieren will, dann kann er mit Hilfe eines Entscheidungsbaums ohne große Fachkenntnis seine Objekte klassifizieren. Bei einem Neuronalen Netz ist das Nachvollziehen der Klassifikation mit einem erheblichen Aufwand verbunden. Nur durch Betrachtung, wie es bei Entscheidungsbäumen sein wird, ist es hier für den Endbenutzer nicht möglich, ein Objekt zu klassifizieren.

Die Effizienz eines Klassifikators wird einerseits dadurch bestimmt, wie sich dieser während der Konstruktion und andererseits während der Anwendung des Modells verhält.

Ein weiteres Kriterium für die Wahl eines Algorithmus ist die Skalierbarkeit für große Datenmengen, wie z.B. dieser zur Anwendung von Hintergrundspeicher-Daten geeignet ist.

Die Robustheit eines Klassifikators sagt etwas darüber aus, wie er auf verrauschte und fehlende Werte reagiert.

Alle genannten Kriterien sollen beachtet und angewendet werden, um einen Algorithmus zu finden, der das Problem der Klassifikation von Körpertypen aus Messdaten löst.

3.2 Merkmale von Körpertypen

Um Körpertypen klassifizieren zu können, müssen die Körpertypen erstmal festgelegt werden. Das Problem dabei ist, dass die Einteilung teilweise subjektiv ist. Unterschiedliche Schneider, Designer und Modellierer haben unterschiedliche Ansichten, wann ein Hohlkreuz beginnt oder ab welchem Bauchumfang eine Person als dick bezeichnet wird. Eine weitere Sache sind die unterschiedlichen Kleidungsstücke, die vom selben Designer konstruiert werden. Für die Konstruktion eines Mantels werden vielleicht nur 3 Körpertypen gebraucht, während für eine Bluse 7 Typen benötigt werden, weil hier und da noch ein Schnitt oder Abnäher extra geschnitten werden muss.

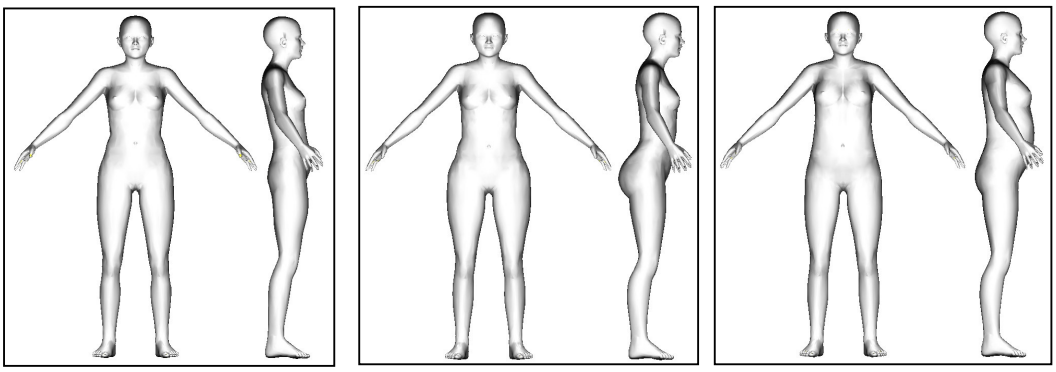

Abbildung 13: Körpertypen – 3D-Modelle

In diesem Projekt wurden 7 unterschiedliche Körpertypen untersucht. Diese werden in der Literatur für Schnittkonstruktionen wie [Stieg92] und in Fachzeitschriften wie [Schnitt] beschrieben, um die Änderungen an den Schnitten zu erklären. Für diese Typen wurden zum Trainieren des Klassifikators virtuelle 3D-Modelle geformt. Diese Modelle haben die Bezeichnungen:

- Mannequinfigur (siehe Abbildung 13 Links)
- Reitergesäßfigur (siehe Abbildung 13 Mitte)
- Bauchfigur/Leibfigur (siehe Abbildung 13 Rechts)
- Hohlkreuzfigur
- Athletenfigur
- Rundrückenfigur
- Normalfigur

Im Folgenden sollen diese Typen beschrieben werden, und vor allem werden die für die Klassifizierung wichtigen Körpermaße ermittelt. Diese Maße bilden dann den Merkmalsvektor. Damit die Maße für unterschiedlich große Frauen vergleichbar sind, werden alle Maße, bevor sie zu Merkmalen gemacht werden, in Relation zur Körperhöhe gebracht.

Für einige Typen müssen neue Maße eingeführt werden, welche ein Schneider per Hand kaum messen kann, aber mit den Augen teilweise gut erkennt. Es geht um die Rückenform, unter anderem für die Mannequinfigur und die Hohlkreuzfigur. Dafür wird eine Hilfslinie hinter den Rücken aus der Seitenansicht eingeführt und als Rückenabstandslinie, kurz AL, definiert. Diese Linie sollte einen Abstand von 1 cm von der am weitesten nach hinten reichenden Körperpartie zwischen Hals und Gesäß haben. Nun können verschiedene Maße in der Rückenform gemessen werden, z.B. der Abstand zwischen der Rückenabstandslinie und dem Rückenpunkt, wo der Gesäßumfang (GU) gemessen wird, kurz AL-G. In den folgenden Beispielen wird das deutlicher.

Tabelle 1: Maßabkürzungen

Kurzbezeichnung	**Name**
KH	Körperhöhe
BU-w	Brustumfang waagerecht
TU	Taillenumfang
HU	Hüftumfang
GU	Gesäßumfang
GD-f	Gesäßdurchmesser frontal

AL-B	Abstand Linie-Brust schräg
AL-BW	Abstand Linie-Brust waagerecht
AL-UB	Abstand Linie-Unterbrust
AL-T	Abstand Linie-Taille
AL-H	Abstand Linie-Hüfte
AL-G	Abstand Linie-Gesäß

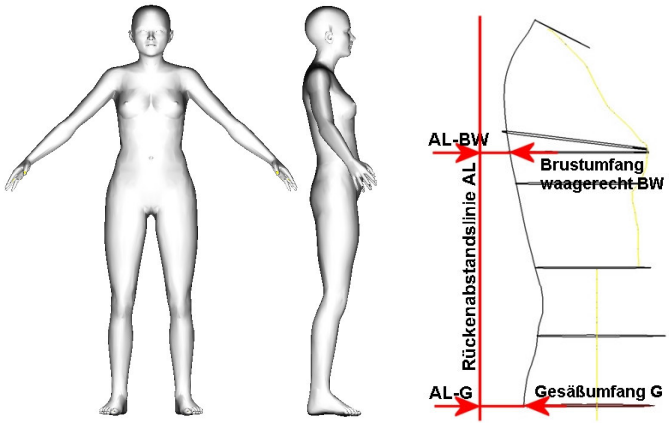

Abbildung 14: Mannequinfigur

In Abbildung 14 sehen wir eine Körperhaltung, die sehr häufig bei jungen schlanken Damen zu finden ist. Man bezeichnet sie auch als Mannequinhaltung, bei der die untere Körperpartie nach vorne geschoben wird, bzw. der obere hintere Rücken über das Gesäß hinaus reicht. In Vergleichen mit anderen Körpertypen hat sich gezeigt, dass dies ein relativ eindeutiges Merkmal ist. Benötigt wird dafür der Abstand Linie-Gesäß AL-G und der Abstand Linie-Brust waagerecht AL-BW. Für das zweite Maß wäre auch theoretisch AL-Unterbrust (UB) möglich. AL-UB könnte aber zu ungenaueren Ergebnissen führen, weil das Maß tiefer gemessen wird und der relative Differenz zwischen AL-G und AL-UB zu gering ausfallen könnte.

Damit die Klassifikatoren nicht zu viele Merkmale berechnen müssen (Performance), kann die Dimension des Merkmalsvektors verringert werden, indem ein Verhältnis der beiden Werte erzeugt wird. Für die automatische Klassifizierung wird dies kein Einfluss auf das Ergebnis haben. So wird aus zwei Merkmalen nur noch eins:

$$\frac{AL-G}{AL-BW}$$

Wenn das Ergebnis > 1 ist, kann man davon ausgehen, dass es sich um eine Mannequinfigur handelt. Bei den Tests mit realen Messdaten zeigte sich, dass auch Personen mit einer Bauchfigur dieses Kriterium erfüllen. Aber durch deren Merkmale, die später

erklärt werden, können diese beiden Personengruppen auch auseinander gehalten werden.

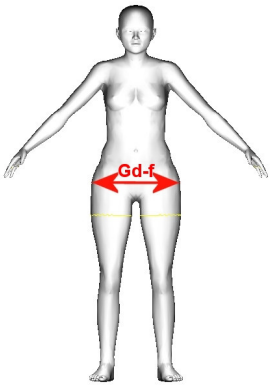

Abbildung 15: Reitergesäßfigur

Das Reitergesäß (Abbildung 15) findet sich häufiger bei älteren Frauen. Das Gesäß wirkt von vorne breiter als bei normalen Figuren. So reicht als Merkmal der Gesäßdurchmesser frontal GD-f, um den Typ von anderen zu unterscheiden. Es kann sein, dass bei einigen Probandinnen die Reiterschenkel etwas tiefer liegen, dann sollte der Durchmesser an der breitesten Stelle bei geschlossenen Beinen gemessen werden.

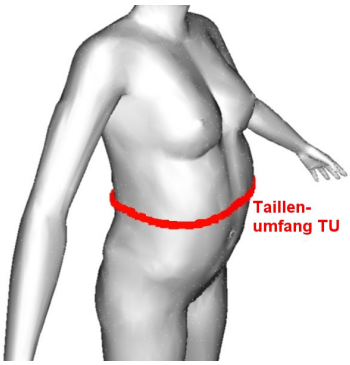

Abbildung 16: Bauchfigur

Bei der Bauchfigur (Abbildung 16) ist eindeutig der Taillenumfang TU das eindeutigste Merkmal zur Unterscheidung von anderen Körpertypen.

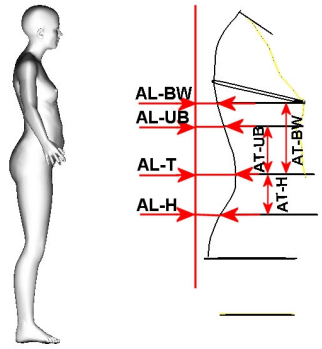

Abbildung 17: Hohlkreuzfigur

Das markante Zeichen der Hohlkreuzfigur (Abbildung 17) ist die Rückenform auf Taillenhöhe, die sehr weit nach vorne ausgeprägt ist. Also werden die Maße für die Taille mit den Nachbarn verglichen. Als Merkmale ergeben sich: Abstand Linie (Taille-Unterbrust) / Abstand Taille Unterbrust (AL-(T-UB) / AT-UB), Abstand Linie (Taille-Hüfte) / Abstand Taille Hüfte (AL-(T-H) / AT-H) und Abstand Linie (Taille-Brust waagerecht) / Abstand Taille Brust waagerecht (AL-(T-BW) / AT-BW). Der erste Teil eines jeden Maßes AL-(X-Y) steht für die Differenz der beiden Maße AL-X **-** AL-Y. Das Ergebnis wird dann dividiert durch den Höhenunterschied, um eine Relation zu erstellen.

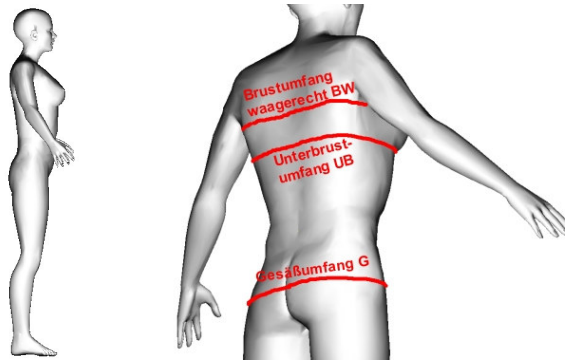

Abbildung 18: Athletenfigur

Die Athletenfigur (Abbildung 18) ist dadurch gekennzeichnet, dass der Oberkörper einen relativ zur Normalfigur größeren Umfang hat als der Hüftumfang HU. Beim Oberkörper sind dabei 2 Maße interessant, der Brustumfang waagerecht und der Unterbrustumfang. Allerdings wird der Brustumfang waagerecht BU-w auch von der Brustgröße beeinflusst, weshalb der Unterbrustumfang UBU genommen werden sollte. Das daraus resultierende Merkmal lautet UBU/HU.

Für die Rundrückenfigur könnten die Maße Abstand Linie-Hals, Abstand Linie Brustumfang waagerecht, Rückenbreite, Vordere Breite und Oberarmdurchmesser genutzt werden. Aus Mangel an guten Daten wurde diese Analyse auf später verschoben.

Alle Personen, die nicht in einer der oberen Klasse einsortiert werden konnten, werden als Normalfigur klassifiziert. Diese Personengruppe sollte auch später den Großteil aller Klassifizierungen ausmachen.

Tabelle 2: Körpertypen und Merkmale

Körpertyp	Merkmale
Mannequinfigur	Abstand Linie-Gesäß / Abstand Linie-Brust waagerecht
Reitergesäßfigur	Gesäßdurchmesser frontal
Bauchfigur	Taillenumfang
Hohlkreuzfigur	Abstand Linie (Taille-Unterbrust) / Abstand Taille-Unterbrust Abstand Linie (Taille-Hüfte) / Abstand Taille-Hüfte Abstand Linie (Taille-Brust waagerecht) / Abstand Taille-Brust waagerecht
Athletenfigur	Unterbrustumfang / Hüftumfang

Die Maße, die an den Personen gemessen werden müssen, sind Körperhöhe, Unterbrustumfang, Taillenumfang, Hüftumfang, Abstand Linie-Brust waagerecht, Abstand Linie-Unterbrust, Abstand Linie-Taille, Abstand Linie-Hüfte, Abstand Linie-Gesäß, Brust Waagerecht-Höhe, Unterbrusthöhe, Taillenhöhe und Hüfthöhe. Aus diesen Maßen können vor der Klassifizierung die Merkmale hergeleitet bzw. berechnet werden.

3.3 Formalisierte und unscharfe Klassifizierungsanforderungen

Menschen benutzen häufig vage Aussagen wie:

- Die Frau ist groß.
- Die Frau ist dick.
- Die Temperatur wird morgen 20 Grad Celsius sein.
- Die Menschen werden in 100 Jahren größer sein als heute.

Diese Aussagen sind weder wahr noch falsch. Der Wahrheitsgehalt dieser Sätze ist entweder nicht eindeutig klar zu bestimmen oder er ist noch nicht bekannt. Ab welcher Größe ist eine Frau groß? Wenn sie ein 1 cm über dem Durchschnitt liegt oder 10 cm? Jeder Mensch klassifiziert viele Wahrnehmungen für sich anders als andere Personen. Das liegt daran, dass wir unsere Sinneswahrnehmungen nicht quantitativ messen, sondern sofort abstrahieren und eine qualitative Einordnung vornehmen. Wir messen also nicht den Salzgehalt einer Speise, sondern unterteilen in salzig, versalzen, fade, usw.

Diese Form von Aussagen bezeichnet man als unscharf, da die Einschätzung, ob etwas salzig ist, ob ein entgegenkommendes Auto sehr nah ist, von Situation zu Situation variieren kann.

Dagegen ist die Aussage „Die Temperatur wird morgen 20 Grad Celsius sein“ unsicher. Es kann im Sinne der klassischen zweiwertigen Logik gesagt werden, ob diese Aussage wahr oder falsch ist, aber nicht zum jetzigen Zeitpunkt. Mehr zu unsicherem Wissen steht in [Lämm01].

Für die Darstellung unscharfer Aussagen eignen sich Fuzzy-Mengen. Fuzzy kommt aus dem Englischen und bedeutet auch so viel wie ungenau, verschwommen, unscharf. Um auch unscharfe Mengenabgrenzungen mathematisch behandeln zu können, dient die Fuzzy-Logik. Sie ist eine Verallgemeinerung der zweiwertigen (boolschen) Logik, die Wahrheitswerte zwischen WAHR und FALSCH zulässt.

Wann wird eine Person als groß bezeichnet? Die Antwort wird zum einen davon abhängen, wer gefragt wird. Zum anderen werden die meisten Befragten bei der Bewertung einer Körpergröße von 1,70 m nicht mit einem klaren „nicht groß“ oder „groß“ antworten.

- Als relativ sicher gilt, dass eine Person, die über 2 m groß ist, von allen als groß angesehen wird.
- Ein Säugling wird im Gegensatz dazu von allen als nicht groß bezeichnet werden.
- Ein Basketballspieler mit einer Größe von 2,20 m wird eine Person von 1,80 m kaum als groß bezeichnen. Für Kinder wird eine 1,80 m große Person aber sehr wohl groß sein.

Um die Personen zum Begriff groß zu klassifizieren, lässt man auch Zugehörigkeitswerte zwischen 0 und 1 zu:

- Personen, die kleiner als 1 m sind, sind nicht groß und haben somit den Zugehörigkeitswert 0.
- Personen, die über 2 m sind, sind groß und haben den Zugehörigkeitswert 1.
- Personen zwischen 1 m und 2 m bekommen einen Zugehörigkeitswert zwischen 0 und 1.

Der Grad an Zugehörigkeit wird meist durch eine Zugehörigkeitsfunktion μ beschrieben:

$$\mu : X \rightarrow [0,1]$$

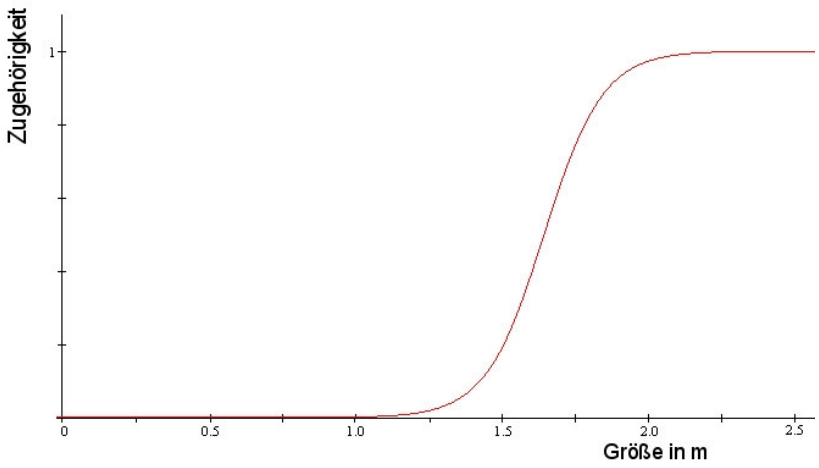

Abbildung 19: Zugehörigkeitsdiagramm

In Abbildung 19 ist eine mögliche Zugehörigkeitsfunktion für große Menschen dargestellt. Eine 1,50 m große Person wird nur zu 20% als groß bzw. zu 80% als nicht groß bezeichnet. Bei einer 1,70 m großen Person liegt die Zugehörigkeit schon bei 0,5.

3.4 Klassische algorithmische Ansätze für die Klassifizierung

Die klassischen Ansätze zur Klassifikation sollen hier nur die näher erläutert werden, die für diese Arbeit untersucht wurden:

- k-Nächste Nachbarn Klassifikation
- Entscheidungsbäume

3.4.1 Nächste-Nachbarn-Klassifikation

Die Nächste-Nachbarn-Klassifikation (Nearest-Neighbour-Classification) gehört zur Klasse der nichtparametrischen Klassifikatoren. Während der Erstellung des Klassifikators werden keine Parameter trainiert und keine Trennfunktionen erzeugt.

Der Klassifikator ist die gesamte Trainingsmenge TR, die abgespeichert wird. So wird bei der Erzeugung des Klassifikators keine Rechenzeit in Anspruch genommen. Aber die Datenmenge von TR ist während der gesamten Klassifikation notwendig.

Das Prinzip bei dieser Klassifikation ist eine Approximation der Verteilungsdichten durch lokale Häufungsgebiete. Um ein neues Muster zu klassifizieren, wird der nächste Nachbar im Merkmalsraum betrachtet. Dessen Klassenzugehörigkeit wird für das neue Objekt übernommen.

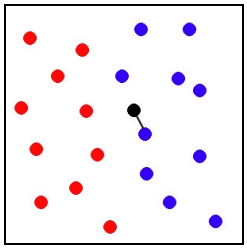

Abbildung 20: Nächste Nachbarn

In Abbildung 20 wird das zu klassifizierende Objekt (schwarzer Punkt) der blauen Klassen zugeordnet, weil sein nächster Nachbar im Merkmalsraum auch ein Objekt aus der blauen Klasse ist.

Bei quantitativen Attributen, wie sie bei den Körpermaßen entstehen, sind die Unterschiede gut vergleichbar. Zum Beispiel ist eine Frau mit einem Taillenumfang von 65 cm dünner als eine mit 90 cm bei gleicher Körperhöhe, da $65-90<0 \Leftrightarrow 65<90$.

Um nun den nächsten Nachbar zu finden, benutzt man Distanzfunktionen, die sich diese Eigenschaft zu Nutze machen. Es gibt viele unterschiedliche Distanzfunktionen, aber die Euklidische Distanzfunktion wird am häufigsten genutzt. Sei $x \in O$, und damit das neue zu klassifizierende Objekt und $y \in TR$ ein bereits bekanntes Objekt, dann berechnet sich der Euklidische Abstand dieser beiden Objekte wie folgt:

$$dist(x,y)=\sqrt{(x_1-y_1)^2+\ldots+(x_d-y_d)^2}=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2} \tag{7}$$

Eine Distanzfunktion ist eine Metrik[4] und muss die folgenden Bedingungen erfüllen:

- $\forall x,y \in O : dist(x,y) \in R^{\geq 0}$
- $dist(x,y)=0 \Leftrightarrow x=y$
- $dist(x,y)=dist(y,x)$
- $\forall x,y,z \in O : dist(x,z) \leq dist(x,y)+dist(y,z)$

Weitere häufig benutzte Distanzfunktionen für quantitative Attribute sind die Manhattan-Distanz:

$$dist(x,y)=|x_1-y_1|+\ldots+|x_d-y_d|=\sum_{i=1}^{d}|x_i-y_i| \tag{8}$$

sowie die Maximumsmetrik:

$$dist(x,y)=\max\left(|x_1-y_1|,\ldots,|x_d-y_d|\right) \tag{9}$$

[4] Metrik - siehe Glossar

Die einfachste Version eines Nächste-Nachbarn-Klassifikators weist ein unbekanntes Objekt x derjenigen Klasse zu, der das von x am wenigsten entfernte Trainingsobjekt y angehört:

$$c(x) = \arg\min_{y \in TR} dist(x, y) \tag{10}$$

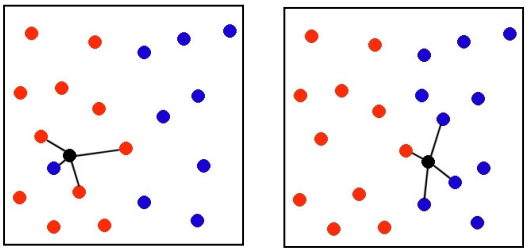

Abbildung 21: Nächste Nachbarn Problem

Wie man in der Abbildung 21 erkennen kann, sollte man nicht nur den einen nächsten Nachbarn betrachten. Im linken Bild ist das neue Objekt (schwarz) am nächsten zu einem Objekt der blauen Klasse. Aber dieses blaue Objekt könnte eine Fehlmessung (Rauschsignal) sein, weil es inmitten der roten Klasse ist. Durch diese Zuordnung wird das Rauschen der Klassifizierung nur stärker. Ähnlich sieht es im rechten Bild aus, dort ist das neue Objekt in der Grenzregion besonders einem roten Objekt nahe. Dieses rote Objekt ist aber nur ein besonders extremer Grenzpunkt, hinter dem sich kein rotes Objekt mehr befinden sollte. Trotzdem wird das neue Objekt ihm zugeordnet.

Um diese Fehlklassifizierung zu verhindern, betrachtet man nicht nur einen Nachbar sondern die k-Nächsten-Nachbarn. Wenn die k-Nächsten-Nachbarn ermittelt wurden, wird das neue Objekt x der Klasse zugeordnet, zu der die meisten Nachbarn gehören. Sei

$$\delta(a,b) = \begin{cases} 0, wenn\ a\ != b \\ 1, wenn\ a = b \end{cases}$$

dann berechnet sich die Klassenzugehörigkeit durch:

$$c(x) = \arg\max_{C_i \in C} \sum_{j=1}^{k} \delta(C_i, c(y_j)) \tag{11}$$

Wenn es vorkommt, dass die maximale Anzahl von Nachbarn von zwei Klassen erreicht wird, dann wird die Klasse gewählt, deren Objekte den durchschnittlich geringsten Abstand haben. Die k nächsten Nachbarn bilden die Entscheidungsmenge.

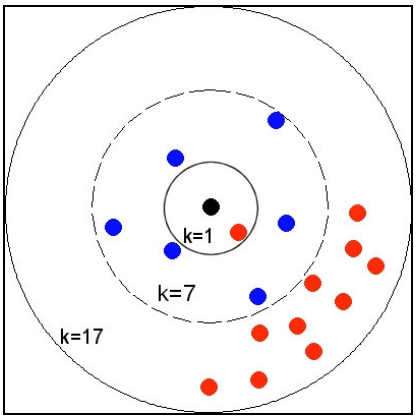

Abbildung 22: k-Nächste-Nachbarn

Die Größe von k ist von großer Bedeutung. In der Abbildung 22 sieht man, was passiert, wenn k zu klein oder zu groß ist. Bei zu kleinem k könnten neue Objekte auf verrauschte Objekte stoßen und so falsch klassifiziert werden und das Rauschen so vergrößern. Bei zu großen k, wie k=17, könnten zu viele Objekte aus anderen Klassen die Entscheidungsmenge beeinflussen, und Klassen mit nur wenigen Objekten gehen „verloren." Um eine gute Klassifizierung zu erreichen, ist ein Wert von k zwischen 3 und 10 am besten geeignet. Sehr häufig wird k=7 benutzt.

Neben der Standardregel „Wähle die Mehrheitsklasse in der Entscheidungsmenge" gibt es noch gewichtete Entscheidungsregeln. Dabei werden die Klassen nach bestimmten Kriterien gewichtet.

Ein Kriterium ist zum Beispiel, alle Objekte in der Entscheidungsmenge nach ihrer Distanz zum gesuchten Objekt zu gewichten. So haben weiter entfernte Objekte weniger Einfluss auf das Ergebnis als weniger entfernte. Dazu benutzt man eine Gewichtsfunktion, beispielsweise:

$$w(x, y) = dist(x, y)^{-2}$$

Die Klassifizierungsformel sieht dann so aus:

$$c(x) = \arg\max_{C_i \in C} \sum_{j=1}^{k} w(x, y_j)\, \delta(C_i, c(y_j)) \qquad (12)$$

Eine andere Möglichkeit wäre, die Entscheidungsregel nach der Verteilung der Klassen zu richten. Klassen, mit weniger als k/2 Objekten in der Trainingsmenge, erhalten keine Chance, ausgewählt zu werden, selbst bei optimaler Distanzfunktion. Zum Beispiel sind 95% aller Objekte in der Trainingsmenge von der Klasse A und nur 5% aus der Klasse B. Aber in der Entscheidungsmenge befinden sich {A, A, A, A, B, B, B}. Damit würde das neue Objekt nach der Standardregel zur Klasse A zugeordnet, aber nach der gewichteten Regel zur Klasse B.

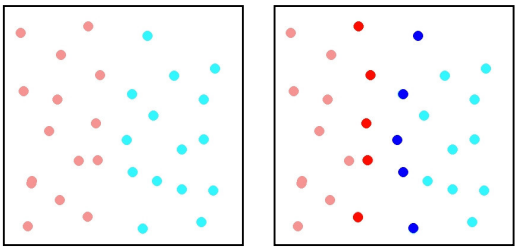

Abbildung 23: Verdichtung

Zur Klassifikation ist es erstmal notwendig, die ganze Stichprobe (Trainingsmenge) zu speichern. Aber durch Verdichtung der Stichprobe kann diese verringert werden. Verdichtung bedeutet hier, unwichtige Muster werden gelöscht. Unwichtige sollten keinen Einfluss auf die Klassifikation haben: Wichtige Muster sind an den Grenzen der Klassen; unwichtige sind weit entfernt der Grenzen (Abbildung 23). Jedes gelöschte Objekt bedeutet aber auch einen Informationsverlust. Bei kleinen Stichproben droht so die Gefahr der Erhöhung der Fehlklassifikation.

Ein Vorteil des kNN-Algorithmus ist seine doch recht leichte Anwendbarkeit, als Eingabe benötigt er nur die Trainingsmenge. Die Trainingsmenge ist der Klassifikator, dieser kann ohne Probleme mit neuen Objekten erweitert werden, die die Klassifizierungsgenauigkeit weiter erhöhen können.

Ein Nachteil dieses Konzepts ist, dass die kompletten Berechnungen erst zur Klassifikationszeit und für jedes neue Objekt neu durchgeführt werden, was bei vielen Trainingsdaten und/oder vielen Attributen zu Geschwindigkeitsproblemen führen kann.

Wenn die Daten viele Attribute haben, die für die Klassifikation keine Bedeutung haben, kann es passieren, dass Objekte gleicher Klasse räumlich weit voneinander entfernt sind. Dadurch verringert sich dann die Klassifikationsgenauigkeit.

Das k-Nächste-Nachbar-Modell liefert kein explizites Wissen über die Klassen.

3.4.2 Entscheidungsbäume

Ein Entscheidungsbaum (decision tree) ist eine spezielle Darstellungsform, bei der ein Baum durch hierarchische Entscheidungen als Klassifikationssystem dient. Der große Vorteil von Entscheidungsbäumen (EB) ist, dass sie für die meisten Benutzer leicht verständlich und lesbar sind.

Den Ausgangspunkt bei einer Klassifikation innerhalb eines EB bildet die Wurzel (root). Man folgt abwärts bis zum nächsten Knoten (node), bei dem ein Test auf eines der Attribute durchgeführt wird. Alle folgenden Kanten stellen die möglichen Ergebnisse dieser Tests dar. Es muss eine Entscheidung gefällt werden, welcher Kante man folgt. Dieses Prinzip wird solange wiederholt, bis man schließlich in einem Blatt (terminal node) endet. Das Blatt stellt eine der Klassen dar. Eine Klasse kann durch mehrere Blätter repräsentiert werden.

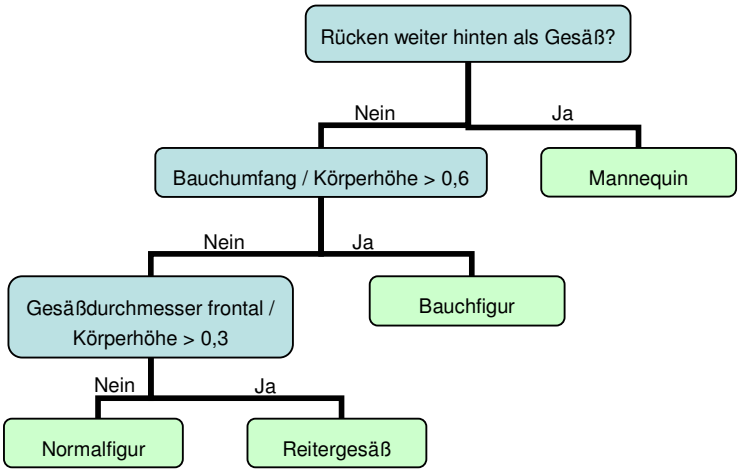

Abbildung 24: Entscheidungsbaum

In Abbildung 24 wird eine mögliche Klassifizierung von Körpertypen durch einen binären EB dargestellt. Bei einem binären EB hat ein Knoten immer nur 2 mögliche Antwortmöglichkeiten. In der Wurzel startet der Prozess der Klassifizierung. Dabei wird in den Merkmalen eines Datensatzes geschaut, ob die Bedingung erfüllt ist oder nicht. Es wird der jeweiligen Kante zum nächsten Knoten gefolgt, die die Bedingung erfüllt. In dem abschließendem Blatt steht dann, in welcher Klasse der Datensatz eingeordnet wurde, z.B. Bauchfigur.

Ein Entscheidungsbaum B ist durch:

- seine einzelnen, immer feineren Zerlegungen bzw. Knoten $k \in B$ und
- seine Zerlegungsvorschriften (splits) $s \in S$

vollständig beschrieben. Dabei ist S der Raum aller möglichen Zerlegungen von $k \in B$.

Für den Bau von EB existieren zahlreiche Algorithmen, die typischerweise das gleiche Schema verwenden [Pretz03]:

1. Sei k die Wurzel des zu erzeugenden Baumes B und TR die Menge der Trainingsdaten.
2. Suche das Attribut x_i und den dazugehörigen Test s_i, der TR am besten in die disjunkten Teilmengen $TR_1,..,TR_m$ teilt.
3. Spalte TR entsprechend auf und erzeuge für alle TR_j einen Knoten k_j als Sohn von k.
4. Für alle $k_j \in B$: falls alle $x \in TR_j$ der selben Klasse C_i angehören, wird k_j ein Blatt mit der Klassenbezeichnung C_i, andernfalls rekursiv weiter bei 2 mit $k=k_j$ und $TR=TR_j$

Bekannte Algorithmen, die diesen Basisalgorithmus verwenden, sind CART, C4.5 und ID3.

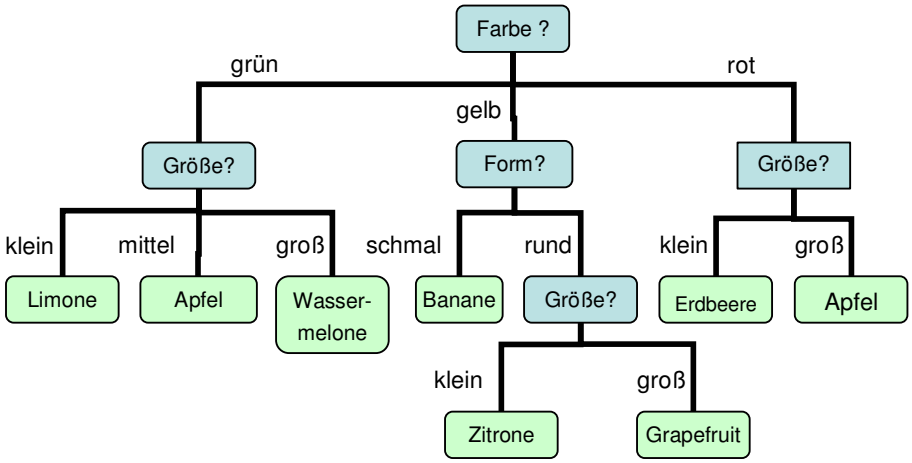

Abbildung 25: Multisplit-Baum

Die Hauptproblematik beim Erzeugen eines Baumes sind die Punkte 2 und 4, für die es auch die unterschiedlichen Ansätze gibt. Wie kann man TR am besten aufteilen? Wie findet man die Zerlegungsvorschriften? Welche splits soll man unter den vorhandenen auswählen? Ab wann ist es nicht mehr sinnvoll, eine TR_i weiter aufzuteilen?

Theoretisch kann bei der Baumerzeugung für jedes Objekt in der Trainingsmenge ein eigenes Blatt erzeugt werden. Dies ist aber nicht sinnvoll, weil der Baum dann übertrainiert ist und ihm so die Generalisierungsfähigkeit genommen wurde.

Es gibt die Möglichkeit, dass ein Split mit nur zwei Verzweigungen erzeugt wird, diese nennt man dann auch binäre Bäume. Bei Bäumen mit mehreren Verzweigungen spricht man von n-ären Bäumen oder auch Multisplits. EB mit beliebigem Verzweigungsfaktor können auf binäre Entscheidungsbäume reduziert werden.

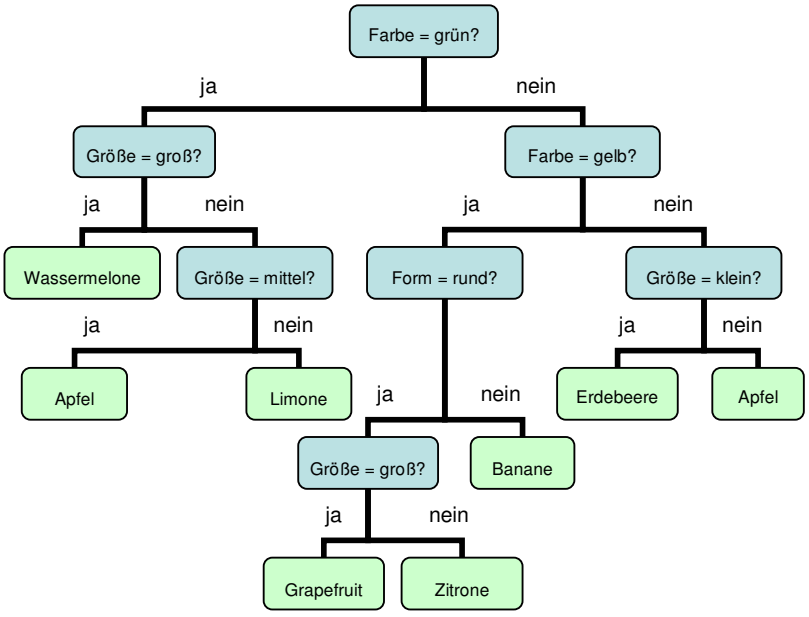

Abbildung 26: Binärbaum

Wie teilt man die Trainingsmenge nun, um einen möglichst guten EB zu erhalten? Intuitiv würde man die Menge so teilen, dass die Objekte der Teilmengen möglichst immer von derselben Klasse stammen. Also so, dass die Teilmengen möglichst rein sind.

Ein Maß für die Reinheit ist die Entropie. Der Begriff wurde ursprünglich auf dem Gebiet der Thermodynamik in der Physik bzw. Chemie eingeführt. In der Mathematik, Statistik und Informationstheorie ist die Entropie ein Maß für die Menge an Zufall, der in einem Zufallsprozess steckt. Ein System mit Entropie 0 ist perfekt geordnet. Wenn die Entropie etwa einen Wert von *1* hat, gilt die Information als zufällig.

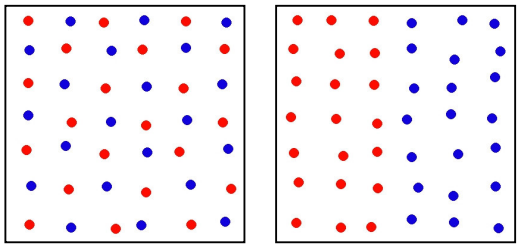

Abbildung 27: Entropie 1

In der Abbildung 27 sind zwei Mengen/Systeme abgebildet, in denen es jeweils 2 Klassen gibt. Bei zufälliger Wahl eines Objekts aus der Menge im linken Bild ist es nicht vorhersehbar, welche Farbe das Objekt hat. Auch im rechten Bild ist es so, obwohl es geordnet aussieht. Aber wird das Bild als eine Menge betrachtet, bei gleicher Anzahl von roten und blauen Objekten, liegt die Wahl einer Farbe bei 50 Prozent. Die Mengen sind unrein und haben somit eine Entropie von 1.

Die Aufgabe besteht nun darin, die Menge so zu teilen, dass eine möglichst „reine“ Menge mit Objekten nur noch einer Klasse vorkommt. Im linken Bild von Abbildung 28 wurde die Gesamtmenge in 2 Teilmengen getrennt, so dass beide Teilmengen absolut rein sind. Die Entropie liegt in diesem Fall für jede Teilmenge bei 0. Dieser Fall ist leider nicht sehr häufig. Oft trifft der Fall wie im rechten Bild der gleichen Abbildung auf. Es herrscht eine große Unordnung vor und es ist nicht leicht 2 möglichst reine Teilmengen zu bilden.

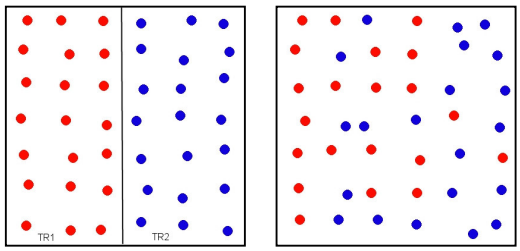

Abbildung 28: Entropie 2

Mathematisch steigt die Entropie, je unreiner die Teilmenge T ist. Für die Erstellung eines Baumes ist die Teilmenge $T \in TR$. Die Variable p_i definiert die Wahrscheinlichkeit, dass ein Objekt $o \in T$ der Klasse C_i angehört. Dies lässt sich mit Hilfe der Trainingsdaten ermitteln durch:

$$p_i = \frac{\left|\{o \in T | o \in C_i\}\right|}{|T|} \tag{13}$$

Dann lässt sich die Entropie einer Teilmenge wie folgt berechnen:

$$entropie(T) = -\sum_{i=1}^{k} p_i \log_2 p_i \tag{14}$$

Um nun einen Split zu bewerten, errechnet man mit Hilfe der Entropie den Informationsgewinn (*information gain*), der die Abnahme der gesamten Entropie durch die Durchführung eines Splits beschreibt:

$$gain(TR, A) = entropie(TR) - \sum_{j=1}^{m} \frac{|TR_j|}{|TR|} enropie(TR_j) \tag{15}$$

Der Parameter A steht für ein Attribut (Merkmal), durch den die Daten gesplittet werden sollen. Das Verfahren nach dem Algorithmus ID3 berechnet den besten Split, indem die Menge TR bei unterschiedlichen Werten des Merkmals A geteilt wird. Für alle möglichen Teilmengen wird der jeweilige Informationsgewinn ermittelt. Es wird der Split gewählt, der den höchsten Wert hat:

$$bestgain(TR, A) = \max_{s \in S}(gain(TR, A)) \tag{16}$$

Die Gefahr besteht, das der Algorithmus Splits bevorzugt, die m Teilmengen mit nur jeweils einem Objekt erzeugt, denn der Informationsgewinn für m Teilmengen mit nur einem Element wäre maximal.

Eine Weiterentwicklung von ID3 ist der Algorithmus C4.5. Dieser verhindert die einelementigen Splits. C4.5 verwendet dafür das Kriterium *gain ratio*, das sich mit Hilfe von *split info* berechnen lässt:

$$split\ info(T, A) = -\sum_{j=1}^{m} \frac{|T_j|}{|T|} \log_2 \frac{|T_j|}{|T|} \tag{17}$$

$$gain\ ratio(TR, A) = \frac{gain(TR, A)}{split\ info(TR, A)} \tag{18}$$

Für einen Test, der m einelementige Teilmengen erzeugt, ist der Wert von split info hoch, der Wert des gain ratio damit niedrig. Um zu vermeiden, dass nun Tests bevorzugt werden, die einen sehr kleinen Wert für split info liefern, setzt C4.5 einen gewissen Schwellenwert für gain(T, s) voraus. Es wird so derjenige Test ausgewählt, dessen Informationsgewinn diesen Schwellenwert erreicht und dessen gain ratio maximal ist:

$$best\ gain\ ratio(TR,s) = \max_{s \in S}\left(gain\ ratio(TR,s)\right) \tag{19}$$

Ein weiterer Algorithmus für die Erzeugung von Entscheidungsbäumen ist der CART-Algorithmus. CART steht für „Classification and Regression Trees“. Ein bedeutendes Merkmal von CART ist, dass nur Binärbäume erzeugt werden können, das heißt, an jeder Verzweigung sind immer genau zwei Äste vorhanden. Das zentrale Element dieses Algorithmus ist also das Finden einer optimalen binären Trennung.

CART benutzt den Gini Diversity Index, der schnell zu berechnen ist und vergleichbar gute Resultate wie das gain ratio liefert. Der Gini-Index ist ein statistisches Maß für Ungleichheit:

$$gini(T) = 1 - \sum_{i=1}^{k} p_i^2 \tag{20}$$

Der Gini-Index einer Partition T_1, bzw. T_2 berechnet sich dann durch:

$$gini(TR) = \sum_{j=1}^{2} \frac{|T_j|}{|T|} gini\left(T_j\right) \tag{21}$$

Der Gini-Index wird umso größer, je größer die Unreinheit der Teilmengen ist. Also ist der beste Split dort durchzuführen, wo der Wert vom Gini-Index am geringsten ist.

Ein Entscheidungsbaum kann solange wachsen, bis alle Trainingsobjekte zum Schluss ein eigenes Blatt repräsentieren. Dann könnten alle Trainingsobjekte auch später immer hundertprozentig richtig klassifiziert werden, aber es entsteht das bereits beschriebene „overfitting“-Problem. Um übertrainierte Bäume zu verhindern, werden Bäume beschnitten. Diese Prozedur wird „pruning“ genannt.

Es werden dabei 2 unterschiedliche Ansätze betrachtet:

1. Prepruning
2. Postpruning

Beim Prepruning wird der Baumwachstum gestoppt, wenn ein bestimmtes Kriterium erfüllt wird:

- Eine bestimmte minimale Anzahl von Objekten befindet sich im Knoten.
- Eine Klasse hat eine festgelegte Dominanz erreicht.
- Eine bestimmte Baumtiefe wurde erreicht.

Bei der zweiten Variante, dem Postpruning, wird der Baum erst komplett aufgebaut und danach beschnitten, dabei werden Teilbäume durch Blätter ersetzt und so der Baum verkleinert.

Postpruning ist das weiter verbreitete Verfahren, denn beim Prepruning besteht die Gefahr, dass der Aufbau des Baumes zu früh abgebrochen wird [Pretz03]. Wenn die Kriterien beim Prepruning zu locker sind, ist es auch nicht sehr effektiv. Aber es lohnt sich, beide Verfahren gleichzeitig zu nutzen. Das Prepruning-Verfahren hat einen Geschwindigkeitsvorteil gegenüber dem Postpruning.

Beim Postpruning werden Teilbäume in Bezug auf ihre Zuverlässigkeit hin untersucht. Teilbäume und Knoten, die keine gute „Performanz“ bieten, werden reduziert [derb00]. Es gibt 2 mögliche Verfahren, die besonders oft genutzt werden:

1. Subtree-Replacement – Unterbaum-Ersetzung und
2. Subtree-Raising – „Hochziehen“ von Unterbäumen.

Das Beschneiden darf nur dann durchgeführt werden, wenn die Fehlerrate dadurch nicht steigt. Da der Baum mit Hilfe der Trainingsdaten aufgebaut wurde, muss mit einer Testmenge die Fehlerrate abgeschätzt werden.

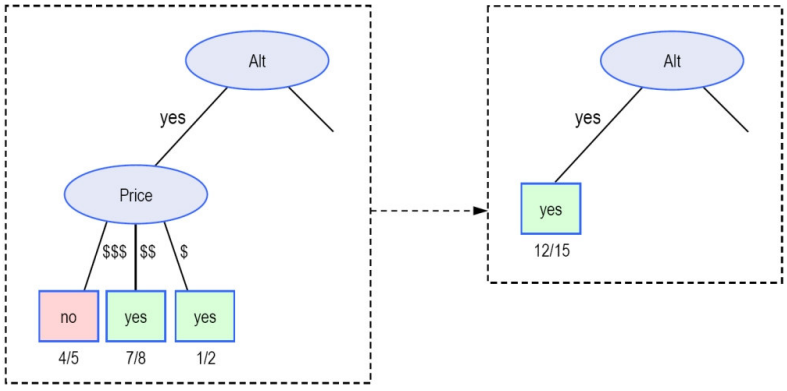

Abbildung 29: Subtree-Replacement

Beim Subtree-Replacement wird ein Teilbaum durch ein Blatt ersetzt (siehe Abbildung 29 - [Köst03]).

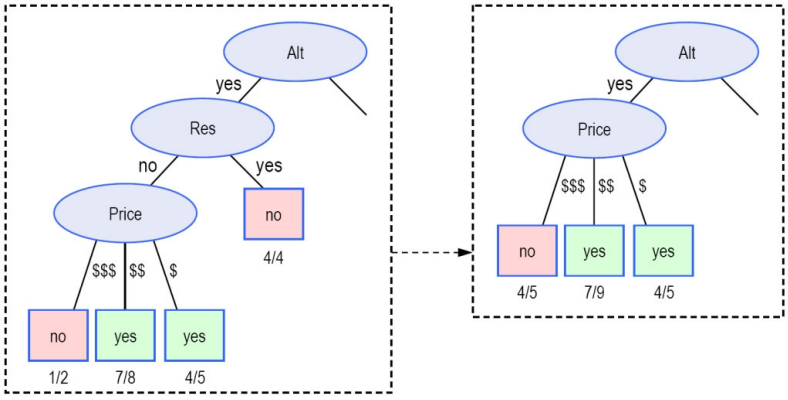

Abbildung 30: Subtree-Raising

Beim Subtree-Raising steigt ein Teilbaum auf eine höhere, der Wurzel nähere Ebene des Entscheidungsbaumes auf (Abbildung 30 - [Köst03]). Dabei wird der eliminierte Vaterknoten in den neuen Knoten des Teilbaumes integriert [Köst03]. Da die Instanzen neu klassifiziert werden müssen, ist dieses Beschneiden doch sehr zeitaufwendig.

Ein großer Vorteil von Entscheidungsbäumen ist, dass diese sehr leicht in Regeln umgesetzt werden können, die ein explizites Wissen über die Klassen liefern. Außerdem ist das Prinzip eines Entscheidungsbaumes leicht verständlich.

3.5 Künstliche Neuronale Netze

Künstliche neuronale Netze sind im Aufbau und in Funktionsweise den biologischen neuronalen Netzen (z.B. dem menschlichen Gehirn) sehr abstrakt nachempfunden. Künstliche neuronale Netze werden nicht nur zur Klassifizierung und Mustererkennung genutzt, sondern auch für anpassungsfähige Software wie virtuelle Agenten und KI-Roboter in Spielen.

Die besondere Eigenschaft neuronaler Netze besteht darin, dass sie komplexe Muster lernen können, ohne dass eine Abstraktion über die diesen Mustern zugrunde liegenden Regeln stattfindet. Sie lernen aus Beispielen, die zu einem Ergebnis im untrainierten Netz führen und berechnen mit Hilfe des Netzfehlers neue Werte für das Netz.

Laut [Call03] ist ein neuronales Netz *„ein Verbund von einfachen Informationsverarbeitungseinheiten, die sich über Gewichte gegenseitig Signale zusenden“* und das, weiter nach [Hoff91], *„dem Prinzip der parallel verteilten Verarbeitung gehorcht.“* Dabei sind die Informationsverarbeitungseinheiten die einzelnen Neuronen und die Gewichte sind Verbindungen zwischen den Neuronen, die jeweils unterschiedlichen Einfluss auf die folgenden Neuronen haben.

3.5.1 Aufbau eines Neuron

Das menschliche Gehirn ist ein Gebilde von rund 10^{10} Nervenzellen, auch Neuronen genannt. Jede dieser Zellen ist mit ungefähr 10^3 bis 10^4 anderen Nervenzellen verbunden, das sind ca. 10^{14} Verbindungen (siehe [Brause91] und [Lämm01]).

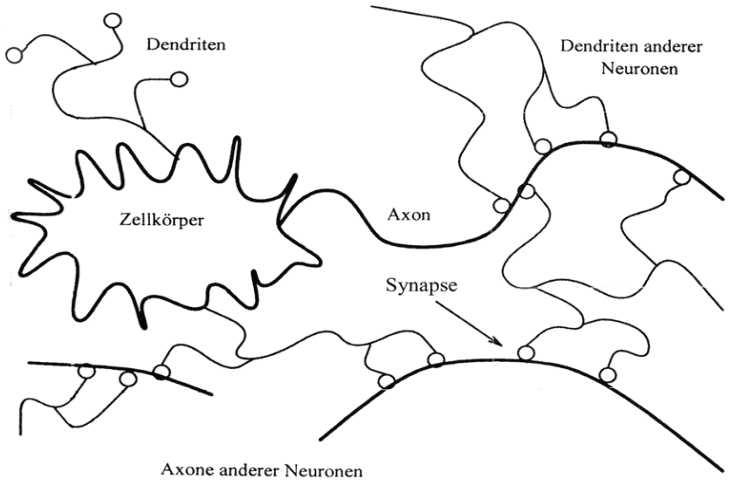

Abbildung 31: Biologische Nervenzelle

Das biologische Neuron besteht im Grundaufbau aus drei Elementen: einem Zellkörper mit dem Zellkern für die Verarbeitung, den Dendriten für die Aufnahme von Erregungen anderer Nervenzellen sowie einem Axon, um die Erregung an andere Neuronen weiterleiten zu können (Abbildung 31 - [Lenz97]). Während von einem Neuron mehrere Dendriten abgehen und sich dann verzweigen, um Informationen aufzunehmen, geht vom Zellkörper nur ein Axon ab, welches den aktuellen Neuronzustand weitergibt. Zwischen den Dendriten und den Axonen befinden sich die Synapsen, die die Informationen stärker oder schwächer weitergeben.

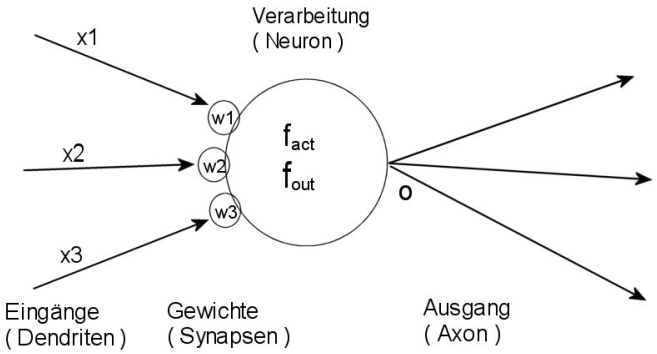

Abbildung 32: Neuronmodell

Wie beim biologischen Gehirn besteht ein künstliches NN aus vielen kleinen aber idealisierten, abstrakten Einheiten, den *Neuronen* (Abbildung 32), in der die Informationsverarbeitung stattfindet. Neuronen sind in diesem Zusammenhang einfache Prozessoren, die sich untereinander mit Hilfe von gerichteten Verbindungen gegenseitig aktivieren. Obwohl die Neuronen nur einfache Verarbeitungseinheiten darstellen, ist es möglich mit ihnen in großer Anzahl komplexe Aufgaben zu bewältigen.

Neuronen können je nach ihrer Funktion in spezielle Schichten eingeteilt werden. Manche Neuronen existieren, um Signale aus der Umgebung zu empfangen und an das Netz weiter zu geben. Diese Eingabe-Neuronen (input neuron) bilden zusammen die Eingabeschicht (input layer).

Die Neuronen, die das Ergebnis der Verarbeitung zurück an die Umgebung übertragen, sind die Ausgabe-Neuronen (output neuron). Die Menge aller Ausgabe-Neuronen sind in der Ausgabeschicht (output layer) zusammengefasst.

Die Einheiten, die weder zur Eingabe- noch zur Ausgabeschicht gehören, sind verdeckte Neuronen (hidden neuron). Sie können auf mehrere verdeckte Schichten (hidden layer) verteilt sein.

In Abbildung 33 sind die Neuronen in einem möglichen Verbund, mit einer Eingabe-, einer Ausgabe- und mehrere verdeckten Schichten, als Neuronales Netz abgebildet.

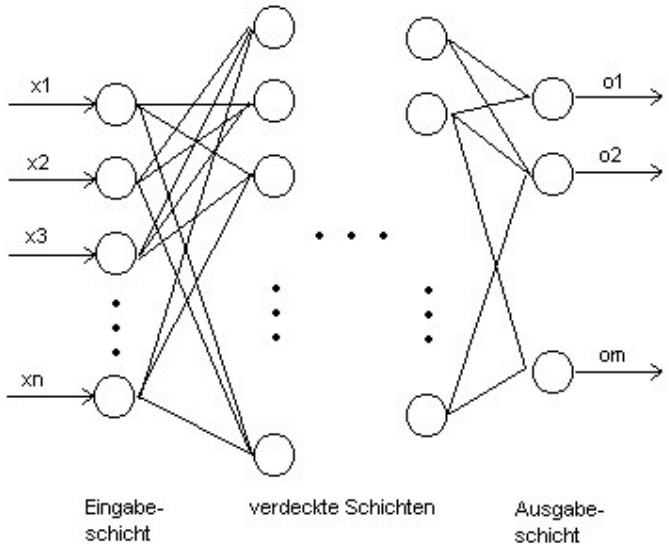

Abbildung 33: Neuronales Netz

Ein Neuron wird bei [Lämm01] wie folgt definiert:

> *„Ein Neuron ist eine Verarbeitungseinheit, die die über die gewichteten Verbindungen eingehenden Werte geeignet zusammenfasst und daraus mittels einer Aktivierungsfunktion unter Beachtung eines Schwellwertes einen Aktivierungszustand ermittelt. Aus dieser Aktivierung bestimmt eine Ausgabefunktion die Ausgabe des Neurons.“*

Ein Neuron N_j führt mehrere Funktionen aus:

- Propagierungsfunktion net_j
- Aktivierungsfunktion f_{act}
- Ausgabefunktion f_{out}

Die *Propagierungsfunktion* f_{prop}, die meist als Netzeingabe net_j bezeichnet wird, verknüpft die Eingabeinformationen x_i mit den dazugehörigen Gewichten w_{ij} zu einer Netzeingabe. Typischerweise werden die Werte durch Aufsummieren kombiniert, wie in (22).

$$f_{prop_j} = net_j = \sum_{i=1}^{n} x_i w_{ij} \tag{22}$$

Es existieren auch noch andere Formen zur Berechnung der Netzeingabe, wie z.B. durch Multiplikation der einzelnen Werte. Eine weitere Möglichkeit ist in [Call03] beschrieben. Die Eingabeneuronen haben normalerweise keine mehrere gewichteten Eingänge, sondern jeweils nur einen ungewichteten Eingang, der ein Messsignal direkt weitergibt.

Die *Aktivierungsfunktion* f_{act} bestimmt aus der Netzeingabe net_j unter Berücksichtigung eines Schwellwertes θ_j die neue Aktivität a_j, den Zustand des Neurons:

$$a_j = f_{act}\left(net_j, \theta_j\right) \tag{23}$$

Die Aktivierung kann Folgendes sein: eine reelle Zahl, die auf ein bestimmtes Intervall eingeschränkt ist (z.B. [0,1]) oder eine diskrete Zahl, wie z.B. {0,1} oder {-1, +1}. Es werden mehrere unterschiedliche Aktivierungsfunktionen eingesetzt, die alle einen mehr oder weniger deutlichen *sigmoiden* Charakter aufweisen. Häufige Verwendung finden lineare Funktionen, Schwellwertfunktionen sowie die logistische Funktion und der Tangens Hyperbolicus (Abbildung 34).

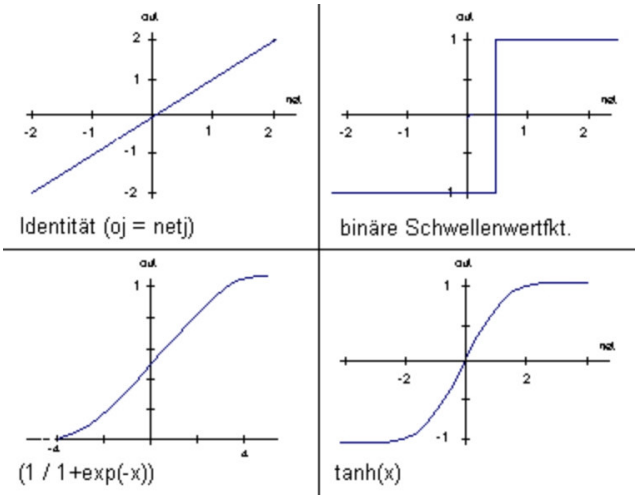

Abbildung 34: Aktivierungsfunktionen

Die Identitätsfunktion ist die Aktivierungsfunktion für die Neuronen auf der Eingabeschicht. Sie gibt das Eingangssignal unverändert weiter. Die Eingabeneuronen geben ihre eingehenden Signale also unverändert in das Netz.

Die binäre Schwellwertfunktion kennt nur 2 Ausgaben, entweder es wird ein Schwellwert θ_j überschritten oder nicht. Als Ausgabemengen können so beispielsweise {0, 1} in (24) oder wie in der Abbildung 34 {-1, +1} eingesetzt werden.

$$f_{act}(net_j, \theta_j) = f_{Schwellwert}(net_j) = \begin{cases} 1, & net_j \geq \theta_j \\ 0, & net_j < \theta_j \end{cases} \tag{24}$$

Im Allgemeinen ist es praktischer, den Schwellwert θ_j bereits von der Netzeingabe abzuziehen. So wird der Schwellwert für die Aktivierungsfunktion zum Wert 0.0 verschoben, dies macht die Anwendung der Aktivierungsfunktion, also auch der folgenden, einfacher ([Lämm01]). Die Schwellwertfunktion ändert sich dann geringfügig zu:

$$f_{act}(net_j) = f_{Schwellwert}(net_j) = \begin{cases} 1, & net_j \geq 0 \\ 0, & net_j < 0 \end{cases} \tag{25}$$

Die Verschiebung ist der Negativwert des Schwellwerts, und in diesem Fall wird die Netzeingabe berechnet durch:

$$f_{prop_j} = net_j = w_{0+} \sum_{i=1}^{n} x_i w_{ij} \tag{26}$$

Die Verschiebung wird so wie eine eingehende Verbindung von einer ständig aktivierten Einheit betrachtet.

Die wohl am häufigsten verwendete Aktivierungsfunktion ist die logistische Funktion:

$$f_{act}(net_j) = f_{\text{logistic}}(net_j) = \frac{1}{1 + e^{-net_j}} \tag{27}$$

Das Ergebnis der Funktion fällt in einen kontinuierlichen Bereich zwischen 0 und 1.

Wenn eine Aktivierung aus dem Bereich [-1, +1] gewünscht ist, wird oft die tanh-Funktion benutzt:

$$f_{act}(net_j) = f_{\tanh}(net_j) = \tanh(x) \tag{28}$$

Es gibt noch viele Variationen von Aktivierungsfunktionen.

Die *Ausgabefunktion* f_{out} berechnet den Ausgabewert o_j aus der Aktivität a_j:

$$o_j = f_{out}(a_j) \tag{29}$$

Im praktischen Einsatz ist meistens die Aktivität a_j auch der Ausgabewert o_j.

3.5.2 Netzarchitekturen

Nachdem die einzelnen Komponenten eines Neurons vorgestellt wurden, folgen nun die Möglichkeiten der Vernetzung mehrerer Neuronen. Dabei kann man die Netzstrukturen in 2 Klassen einteilen:

1. **Netze ohne Rückkopplung**, auch vorwärts gerichtete Netze (feedforward-Netze) genannt,
2. **Netze mit Rückkopplung**, auch als rekurrente Netze bezeichnet.

Ein Netz ohne Rückkopplung ist in verschiedene Schichten aufgebaut mit zumindest einer Eingabe- und Ausgabeschicht und dazwischen liegenden versteckten Schichten in einer variablen Anzahl.

Bei einem feedforward-Netz führt kein Pfad ausgehend von einem Neuron, auch über dazwischen liegende Neuronen, zu demselben Neuron zurück, d.h. die Daten verlaufen nur in einer Richtung von der Eingabe- zur Ausgabeschicht.

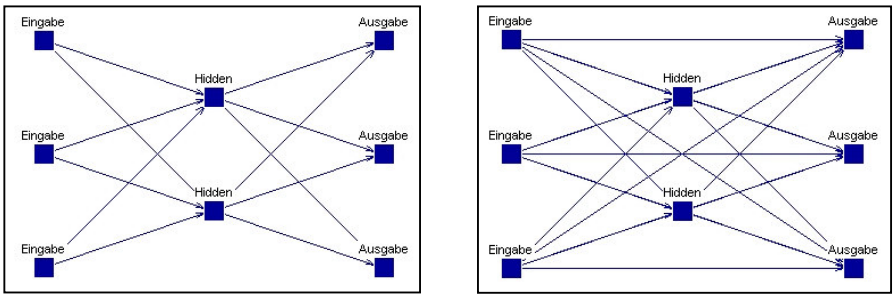

Abbildung 35: Netze ohne Rückkopplung

Man unterscheidet außerdem ebenenweise verbundene und allgemeine feedforward-Netze. Ebenenweise verbundene feedforward-Netze sind in mehrere Schichten eingeteilt, wobei es nur Verbindungen von einer Schicht zur nächsten gibt (Abbildung 35 links). Man spricht von vollständig verbundenen Netzen, falls jedes Neuron der Schicht U_i mit jedem Neuron der darauf folgenden Schicht U_{i+1} verbunden ist.

Allgemeine feedforward-Netze besitzen dagegen auch so genannte shortcut connections, also Verbindungen zwischen Neuronen, die Ebenen überspringen (Abbildung 35 rechts).

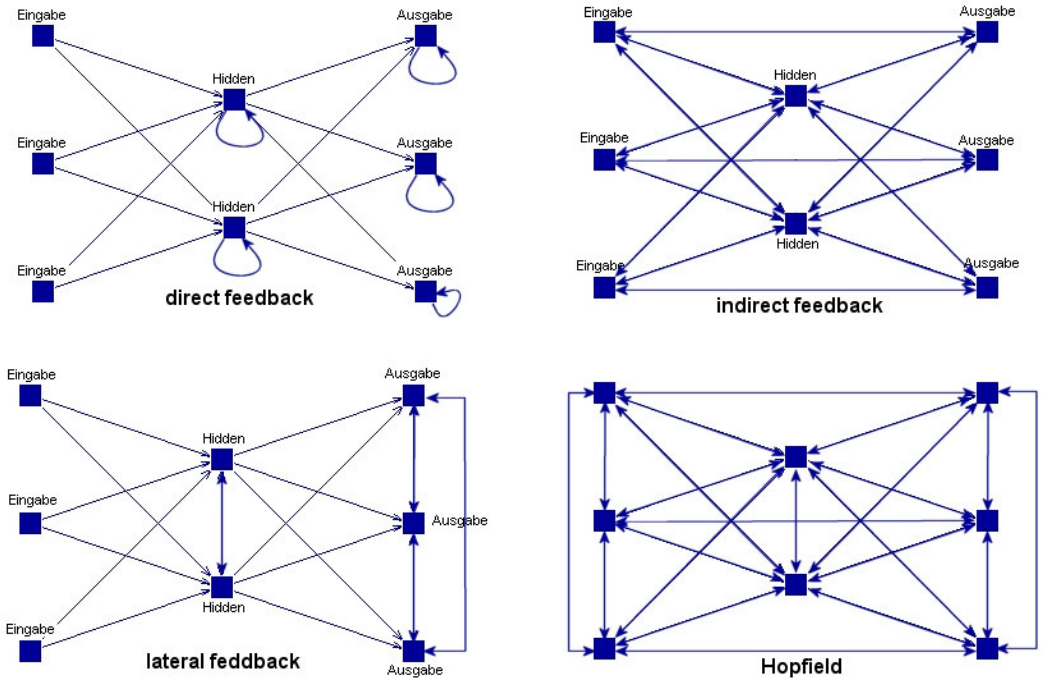

Abbildung 36: Netze mit Rückkopplung

Netze mit Rückkopplungen (Abbildung 36) kann man unterteilen in:

- **Netze mit direkten Rückkopplungen** (direct feedback):

  Die Neuronen haben eine Verbindung von ihrer Ausgabe zurück zur Eingabe, dadurch verstärken oder schwächen sie ihre eigene Aktivierung.

- **Netze mit indirekten Rückkopplungen** (indirect feedback):

  Die Neuronen niedriger Schichten (näher zum InputLayer) erhalten Informationen von Neuronen höherer Schichten (näher bei OutputLayer)

- **Netze mit Rückkopplungen innerhalb einer Schicht** (lateral feedback):

  Die Neuronen innerhalb einer Schicht sind miteinander verbunden.

- **Vollständig verbundene Netze ohne direkte Rückkopplungen** (Hopfield-Netze):

  Alle Neuronen haben mit allen anderen Neuronen, außer sich selbst, Verbindungen in beide Richtungen.

Zur Klassifizierung können theoretisch alle Netzstrukturen gewählt werden, aber diese Arbeit beschränkt sich auf die vorwärts gerichteten Netze. Diese sind einfacher zu handhaben und genügen der Aufgabe zur Einteilung von Körpertypen. Weitere Details zu den Netzen mit Rückkopplungen sind in der entsprechenden Literatur, wie [Lämm01] und [Call03], nachzulesen.

3.5.3 Trainieren eines neuronalen Netzes

Bevor ein neuronales Netz eine Aufgabe lösen kann, muss es zunächst trainiert werden. Wenn ein neuronales Netz lernt, heißt das, die Gewichtswerte werden aufgrund einer bestimmten Lernregel an den Eingangsverbindungen auf sinnvolle Weise angepasst.

Beim Lernen wird in verschiedene Lernparadigmen, wie in [Lämm01], unterschieden:

- **Überwachtes Lernen (supervised learning):**

  Beim überwachten Lernen sind zu den Eingabedaten der Trainingsmenge auch die richtigen Ausgabewerte bereits vorgegeben. Die vom neuronalen Netz tatsächlich errechneten Ausgaben werden mit den vorher festgelegten verglichen und anhand der Differenz wird das Netz entsprechend angepasst, d.h. die Gewichtswerte werden verändert. Der Fehler wird immer kleiner. Die Trainingsdaten sollten möglichst alle Aspekte des Problems abdecken.

  Das Netz kann somit nach dem Lernvorgang Einsatz finden, um neue Eingangsdaten entsprechend der trainierten Regeln Ausgaben zu klassifizieren.

  Diese Lernmethode wird in der vorliegenden Arbeit verwendet, um das neuronale Netz zu trainieren, das aus den Messwerten der Körper (Eingangsdaten) die zugehörigen Körpertypen (Ausgangsdaten) auswählen soll.

- **Bestärktes Lernen (reinforcement learning):**

  Der Unterschied zum überwachten Lernen besteht darin, dass für die Ausgabe des Netzes nur mitgeteilt wird, ob diese richtig oder falsch ist. Einen genauen Wert über die Differenz erhält das Netz dabei nicht.

- **Unüberwachtes Lernen (unsupervised learning):**

  Beim unüberwachten Lernen versucht das Netz selbst ohne äußere Beeinflussung die Ausgabedaten in Ähnlichkeitsklassen aufzuteilen.

Es gibt unterschiedliche Lernalgorithmen, mit denen ein neuronales Netz trainiert werden kann. Eine einfache und dennoch gute Regel für die Gewichtsanpassung, also dem Trainieren des Netzes, ist unter dem Namen Backpropagation bekannt. Der Backpropagation-Algorithmus durchläuft ein Netz zweimal, einmal vorwärts von der Eingabeschicht zur Ausgabeschicht und anschließend rückwärts von der Ausgabeschicht zur Eingabeschicht.

Zu Beginn des Netztrainings werden die Gewichte mit zufälligen niedrigen Werten initialisiert, z.B. zwischen -0,3 und +0,3 [Call03].

Beim Vorwärtsdurchlauf werden die Merkmalsvektoren $\vec{x}$ der Trainingsdaten TR an die Eingabeeinheiten übergeben. Bei quantitativen Merkmalen muss so die Eingabeschicht mindestens genauso viele Neuronen besitzen wie die Daten Merkmale haben. Das Netz berechnet dann die möglichen Ausgaben an der Ausgabeschicht. Um die Ausgabe des Netzes zu ermitteln, müssen der Reihe nach die Ausgaben der Eingabeneuro-

nen, verdeckten Neuronen und Ausgabeneuronen berechnet werden. Diese Berechnung findet auch nach dem Trainieren des Netzes immer wieder statt und wird, wie in 3.5.1 „Aufbau eines Neuron“ beschrieben, durchgeführt.

An jedem Ausgabeneuron o_j liegt nun ein Wert an, dieser wird mit dem erwarteten Wert t_j (teaching output) verglichen, in dem die Differenz der beiden Werte gebildet wird:

$$f_j = t_j - o_j \tag{30}$$

Aus dem resultierenden Fehlerwert f_j wird für jede Ausgabeeinheit ein Fehlersignal θ_j berechnet. Wird als Aktivierungsfunktion die logistische Funktion verwendet, berechnet sich das Fehlersignal wie folgt:

$$\delta_j = \begin{cases} o_j \cdot (1 - o_j) \cdot f_j, & \text{falls j Ausgabe - Neuron} \\ \mathrm{o_j} \cdot (1 - o_j) \cdot \sum_k \delta_k \cdot w_{jk}, & \text{falls j Hidden - Neuron} \end{cases} \tag{31}$$

Wird statt der logistischen Funktion der Tangens Hyperbolicus als Aktivierungsfunktion benutzt, ergibt sich folgendes:

$$\delta_j = \begin{cases} (1 - \tanh^2 o_j) \cdot f_j, & \text{falls j Ausgabe - Neuron} \\ (1 - \tanh^2 o_j) \cdot \sum_k \delta_k \cdot w_{jk}, & \text{falls j Hidden - Neuron} \end{cases} \tag{32}$$

Die Fehlersignale der versteckten Neuronen werden anhand der Fehlersignale δ_k aller k-nachfolgenden Neuronen und den zugehörigen Gewichten w_{jk} berechnet.

Die Herleitung dieser Formel wird sehr gut und ausführlich in den Büchern von [Lämm01] und [Call03] beschrieben.

Als letzter Schritt für ein Muster müssen für alle Schichten die Gewichte jeder Einheit neu berechnet werden. Die neuen Werte werden mit Hilfe der einzelnen Fehlersignale und einer vorher festgelegten Lernrate η aktualisiert. Die Lernrate η gibt an, um wie viel das Gewicht angepasst wird. Das neue Gewicht w'_{ij} berechnet sich durch:

$$w'_{ij} = w_{ij} + \eta \cdot o_j \cdot \delta_j \tag{33}$$

Dieser Durchlauf wird für alle Trainingsdaten mehrmals durchgeführt. Entweder so oft, wie es vorher festgelegt wurde, oder bis der Gesamtfehler bei einem Durchlauf mit allen Mustern geringer ist als ein vorher festgelegter Toleranzwert. Der Gesamtfehlerwert E eines Durchlaufes berechnet sich aus den einzelnen Fehlerwerten f_j. Eine der möglichen Varianten zur Berechnung ist in (34) dargestellt.

$$E = \sum (t_j - o_j)^2 \tag{34}$$

Eine grundsätzliche Kritik gegen die Anwendung neuronaler Netze ist, dass ein neuronales Netz häufig als Black Box angesehen wird, deren Funktionsweise unverstanden bleibt. Aber dann muss man auch die Frage stellen: „Muss ein Endbenutzer des trainierten Netzes die Funktionsweise verstehen?“ Soll er nachvollziehen können, warum das Netz ein Objekt in diese und nicht in jene Klasse eingeordnet hat? Diese Entscheidung muss bei der Wahl eines Klassifizierungsalgorithmus mit einfließen.

4 Algorithmisch-programmtechnischer Teil

4.1 Algorithmisches Konzept zur Klassifizierung von Körpertypen nach Anwendervorgabe

Die Entwicklung einer Konzeption ist nötig, um aus den gestellten Anforderungen sowie den erläuterten Grundlagen ein Gesamtkonzept zu entwickeln. Dies ermöglicht es, den Überblick über die Komplexität der Aufgabe zu behalten. Das erarbeitete Konzept soll die grundlegenden Designentscheidungen widerspiegeln und so als Wegweiser für die spätere Implementierung dienen.

Für die Realisierung einer Klassifikation von Körpertypen ist es notwendig, diese miteinander zu vergleichen. Der theoretische Vergleich zwischen drei ausgewählten Klassifizierungsalgorithmen wurde im Kapitel 3 durchgeführt. Um diese Algorithmen auch im praktischen Einsatz zu testen, werden 3 Prototypen von Programmen erzeugt, die jeweils eine Klassifizierungstechnik benutzten. Die Prototypen heißen in der vorliegenden Realisierung kNearestNeighbor, DecisionTrees und NeuronalClassify.

Alle drei Programme sollen in ihrer Bedienung und im Ablauf nach dem gleichen Prinzip funktionieren (Abbildung 37). Zu Beginn muss ein Klassifikator trainiert werden. Dafür müssen bereits sortierte Trainingsdaten geladen werden. Dann werden die Merkmale gewählt, die mindestens genutzt werden sollen. Es können auch mehr als die in Kapitel 3.1 beschriebenen Parameter verwendet werden, um zu testen, ob vielleicht andere eine bessere Klassifizierungsgenauigkeit erreichen.

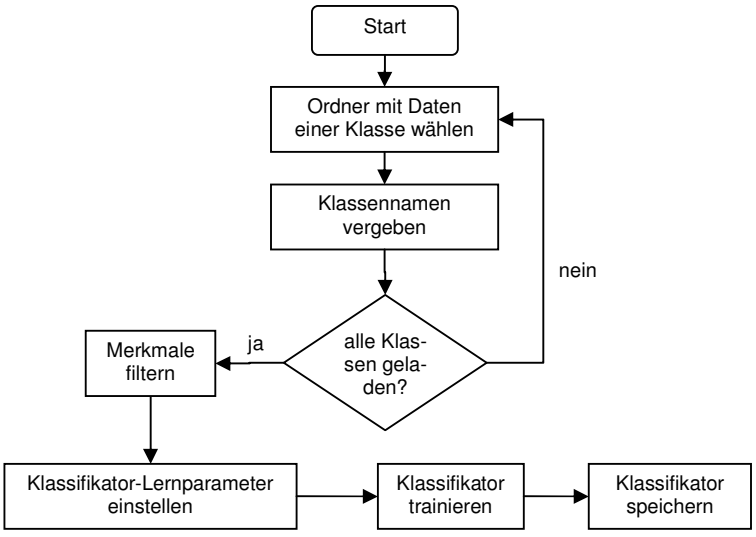

Abbildung 37: Programmablaufplan - Klassifikator erstellen

Nachdem die Parameter ausgesucht worden sind, werden die Daten konvertiert. Dabei werden alle für die Klassifizierung nicht gewählten Parameter aus den Trainingsdaten

gelöscht. Zusätzlich werden die Merkmale durch die Körperhöhe dividiert, um die Relation zur Größe der Person zu berücksichtigen. Nachdem alle Werte entsprechend angepasst wurden, müssen diese für den kNN und das NN in einem Bereich zwischen 0,1 und 0,9 normiert werden.

Um die Klassifikator optimal zu trainieren, sind vorher spezifische Parameter einzustellen. Danach kann der Klassifikator mit den Trainingsdaten trainiert werden.

Damit zu späteren Zeitpunkten auf den Klassifikator zurückgegriffen werden kann, muss dieser in einem für jeden Klassifizierungstyp unterschiedlichen Format gespeichert werden. Zusätzlich müssen beim kNN und NN die Skalierungsbereiche der Daten gespeichert werden, damit die zukünftigen Daten in den richtigen Bereich skaliert werden.

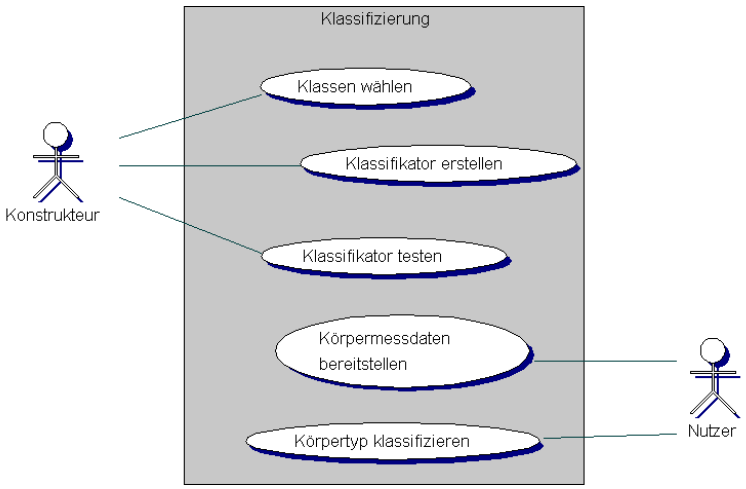

Abbildung 38: Use-Case-Diagramm Klassifizierung

Nach dem ein Klassifikator erstellt wurde, muss dieser auf seine Genauigkeit und Generalisierungsfähigkeit hin überprüft werden. Dazu werden andere Daten eingeladen und nacheinander klassifiziert. Die falsch klassifizierten Daten müssen an den Nutzer zurückgegeben werden, damit dieser dann einschätzen kann, ob der Klassifikator gut genug ist, um die Aufgabe zu lösen. Eventuell muss der Klassifikator mit anderen Lernparametern noch einmal neu trainiert werden, oder die Trainingsdaten müssen noch einmal überarbeitet werden.

In Abbildung 38 wird die Erstellung und Nutzung eines Klassifikators dargestellt. Es gibt zwei Hauptakteure. Ein Konstrukteur, der für ein Design verschiedene Schnitte anlegt und dafür auch die benötigten Klassen wählt. Daraus erstellt er den Klassifikator, den er auch testen sollte.

Die Endnutzung soll dann so sein, dass der trainierte Klassifikator und gegebenenfalls zusätzliche Parameter vom System geladen werden. Nachdem eine Person, der zweite Hauptnutzer, in der Messkabine vermessen wurde, werden die Messdaten in die Klassifizierungsklasse gegeben. Dort werden die wichtigen Daten gefiltert, berechnet und

skaliert. Dann erfolgt die Klassifizierung. Der klassifizierte Körpertyp wird dann einerseits mit den Messwerten zum Hersteller der Kleidung gesendet, und anderseits wird für die virtuelle Anprobe ein individueller 3D-Avatar generiert, der die Bekleidung besser präsentieren kann.

4.2 Integration in bestehende Softwarebibliotheken des Bereichs 3DDV der GFaI

Bei der Implementierung war eine Integration in die bestehende Softwarebibliothek zu gewährleisten, um die Verwendbarkeit und Wartbarkeit zu erleichtern. Dies beinhaltet die Nutzung bestehender Softwaremodule sowie die bestmögliche Kompatibilität der neu implementierten Module.

Die bestehende Klassenbibliothek ist in der Programmiersprache C++ geschrieben. Das Projekt um die Körpermesskabine wurde in der Entwicklungsumgebung Borland C++Builder 6 entwickelt. Um die volle Kompatibilität und Verwendung von objektorientierten Programmiertechniken zu ermöglichen, wurden die zu implementierenden Klassen ebenso in dieser Umgebung entwickelt.

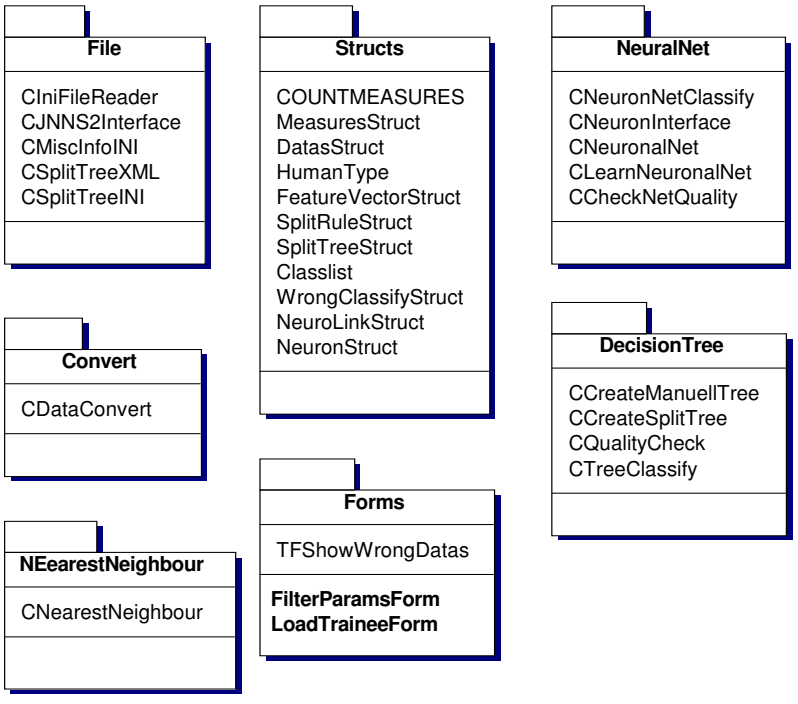

Abbildung 39: Packagediagramm ClassifyLib

Die erzeugten Klassen, die von allen drei Prototypen benutzt werden, sind in einem Verzeichnis als Bibliothek zusammengefasst. Dabei sind die Klassen innerhalb dieser *ClassifyLib* nochmals nach ihrer Benutzung unterteilt (Abbildung 39). Es gibt ein Verzeichnis *File*, in dem sich alle Klassen befinden, die etwas in Dateien hineinschreiben

oder herauslesen. Das Verzeichnis *Structs* beherbergt Strukturen, die neue Datentypen definieren. In *Forms* sind Formulare enthalten, die das Einlesen der Klassen und Auswählen der Parameter ermöglichen. In *Convert* sind zahlreiche Konvertierungsfunktionen enthalten, z.B. die aus Daten einzelner Personen mit anderen Daten den Merkmalsvektor erzeugen.

Die restlichen drei Verzeichnisse *NeuralNet, DecisionTree, NearestNeighbour* beherbergen die entscheidenden Klassen für die Klassifizierung und deren Lernalgorithmen.

4.3 Umsetzung des Konzeptes für die Klassifizierung von Körpertypen

In diesem Abschnitt wird auf die Umsetzungen der verschiedenen Klassifizierungsalgorithmen näher eingegangen. Dazu gehören detaillierte Beschreibungen sowie die kompletten Algorithmen der Verfahren. Weiterhin werden die bei der Umsetzung des Verfahrens aufgetretenen Probleme geschildert und die dafür gefundenen Lösungen vorgestellt.

4.3.1 Datenvorbereitung

Bevor der Klassifikator trainiert werden kann, müssen die Daten vorbereitet werden.

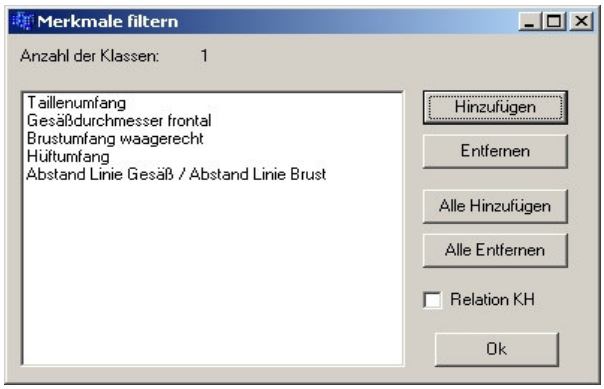

Abbildung 40: Dialogfenster Filterparameter

Um die wichtigen Daten von den unwichtigen zu trennen, gibt es Filterparameter. Diese liegen als Vector von *AnsiStrings* vor. Sie können zum Beispiel mit der Formularklasse *TFFilterParam* erzeugt werden (Abbildung 40). Dabei werden alle bekannten Maßbezeichnungen, die in der *MeasureStructs.cpp* liegen, aufgelistet und zusätzlich werden alle Maßbezeichnungen miteinander in ein Verhältnis gestellt, in dem 2 Bezeichnungen mit ein Slash („ / “) voneinander getrennt werden. So kann bei Bedarf die Dimension des Merkmalsraums reduziert werden. Der Nutzer der den Klassifikator erstellt, muss aus der Liste alle notwendigen Parameter wählen, die für die Klassifizierung in Frage kommen. Beispiele für die Parameter sind „Taillenumfang“, „Gesäßumfang“, „Brustum-

fang / Hüftumfang“ und „Abstand Linie Gesäß / Abstand Linie Brust waagerecht“ (weiteres zur Herleitung der richtigen Parameter in 3.2 „Merkmale von Körpertypen“).

Bevor gefiltert wird, müssen alle Werte durch die Körperhöhe dividiert werden, damit die Größe keinen Einfluss mehr auf die Klassifizierung hat. Dazu gibt es in der Klasse *CDataConvert* die Funktion *convertDatawithKHRelation*.

Dann werden die noch nach Klassen geordneten Messdaten gefiltert nach den Filterparametern. Dabei werden alle einzelnen Messnamen gesucht, aber auch die mit einem Slash verbunden Messwerte werden zusammengefügt durch Division.

Danach müssen die Trainingsdaten mit *convertClasses2FeatureList* in Vektoren konvertiert werden. Dazu gibt es das *Struct* ***FeatureVectorStruct.*** Ein einzelnes Element beschreibt dabei auch nur einen Datensatz. Ein Vector soll den ganzen Merkmalsraum aufspannen. Die wichtigsten Parameter von *FeatureVectorStruct* sind *Measures* und *ClassTyp*. In *Measures* sind alle gefilterten Maße und ihre Bezeichnungen gespeichert. *ClassTyp* ist die Identifizierung für die Klasse. Das Attribut *Datas* speichert vom jeden Datensatz noch zusätzliche Daten, wie *PersonID* zum Wiederauffinden der Originaldateien. *Typname* ist die Bezeichnung der Klasse.

```
typedef struct {
     MeasuresStruct   Measures;
     DatasStruct      Datas;
     unsigned int     ClassTyp;
     AnsiString       TypName
} FeatureVectorStruct;
```

Für kNN und NN müssen alle einzelnen Werte pro Parameter normiert werden, damit kein Parameter den anderen überwiegt. Dafür gibt es die Funktion *calcNewArea* in *CDataConvert*. Diese Skalierinformationen sollten dann für die spätere Benutzung auch auf jeden Fall mit *CMiscInfoINI::SaveData2INI* gespeichert werden.

4.3.2 Klassifikator lernen

Im Folgenden werden die Algorithmen beschrieben, um einen Klassifikator zu trainieren. Der k-Nächste-Nachbar-Klassifikator ist bereits fertig, wenn die Daten in dem letzen Abschnitt beschriebenen *FeatureVectorStruct* vorliegen.

4.3.2.1 Entscheidungsbaum

Bei den Entscheidungsbaumalgorithmen wurden Binärbäume bevorzugt, da sie leicht verständlich bei der Auswertung sind und der Endnutzer die Klassifizierung nachvollziehen kann. Für diese Untersuchung wurde im speziellen der CART-Algorithmus benutzt, da er auch speziell für Binärbäume entwickelt wurde.

Bevor der Klassifikator mit *CCreateSplitTree* erstellt werden kann, müssen bestimmte Parameter festgelegt werden. Folgende Parameter sind so genannte Prepruning-Parameter, also Abbruchkriterien bei der Baumerzeugung: maximale Baumtiefe (Ebenenanzahl), die Reinheit eines Knotens sowie die minimale Anzahl von Elementen in einem Knoten, wenn er nie eine ausreichende Reinheit erhält.

Ein weiterer Parameter ist die Aufteilung einer Menge, die auf Entropie geprüft werden soll. Der Parameter beschreibt, die Anzahl der zu untersuchenden Teilmengen.

Beim Starten der Baumerzeugung muss angegeben werden, ob ein Postpruning durchgeführt werden soll.

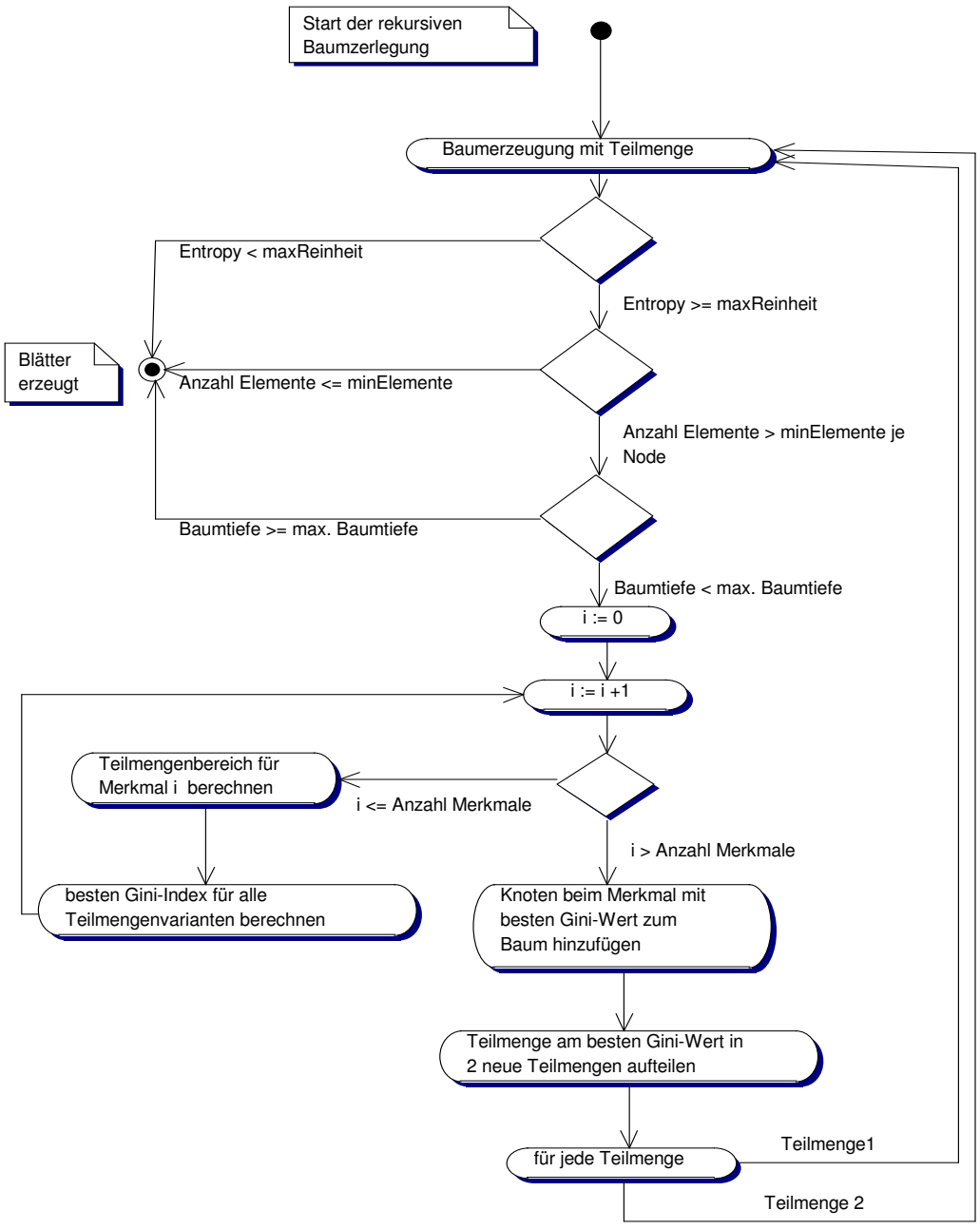

Abbildung 41: Aktivitätsdiagramm CART-Algorithmus

In Abbildung 41 wird der Algorithmus für die Entwicklung eines CART-Entscheidungsbaums dargestellt. In der Klasse *CCreateSplitTreegibt* es dafür die Methode *calcSplitTreeCART*. Der Vector von *FeatureVectorStruct* mit n „*Measures*"-Werten bildet einen n-dimensionalen Raum. Für jedes Merkmal wird die Menge in eine vorher bestimmte Anzahl von unterschiedlichen zweifachen Teilmengen geteilt. Aus allen Teilmengen bei allen Merkmalen wird die beste Reinheit einer Teilmengenaufteilung gesucht. Wenn das Merkmal und die Aufteilung gefunden sind, wird im Baum ein neuer Knoten hinzugefügt, mit dem Merkmal als Regel und der Grenze der beiden Teilmengen als Regelwert. Die beiden Teilmengen werden erneut durch die Funktion verarbeitet, solange bis alle Teilmenge an den Abbruchbedingungen zu Beginn der Methode in Blätter umgewandelt wurden.

Die Entscheidungsbäume haben folgende Struktur:

```
typedef struct {
     int              ID;
     int              FatherID;
     int              LeftNodeID;
     int              RightNodeID;
     int              NodeTyp;
     SplitRuleStruct  SplitRule;
     unsigned int     ClassTyp;
     AnsiString       TypName;
} SplitTreeStruct;


typedef struct {
     AnsiString       FeatureName;
     Double           FeatureValue;
} SplitRuleStruct;
```

Jeder Knoten und jedes Blatt hat eine *ID* zur Identifizierung. Zusätzlich kennt jedes Element seinen Vater (*FatherID*) und seine Nachfolger (*LeftNodeID*, *RightNodeID*). Der Wurzelknoten (root) hat keinen Vater über sich und hat deshalb als *FatherID* den Wert -1, als eigene ID den Wert 0. Die Blätter eines Baumes dagegen haben keine weiteren Nachfolger, deshalb haben diese auch den Wert -1. Der *NodeTyp* fasst dies zusammen, wenn er 0 ist, handelt es sich bei dem Element um ein Blatt, bei 1 um ein Knoten.

In *SplitRule* ist das Merkmal und der Wert gespeichert, an dem sich die Teilmengen trennen sollen. *ClassTyp* und *TypName* sind Index und Bezeichnung der Klasse, deren Trainingsobjekte im Knoten bzw. Blatt überwiegen. Wenn ein Messdatensatz bei einer Klassifizierung in einem Blatt landet, kann sie aus dem Typnamen die Klassenbezeichnung erhalten.

Nachdem ein Entscheidungsbaum erstellt wurde, muss dieser für die spätere Verwendung abgespeichert werden. Für das Speichern eines Baumes wird das INI-Dateiformat benutzt. Im INI-Format werden häufig Konfigurationen abgespeichert, ihr

Vorteil liegt in der leichten Interpretierbarkeit für Computer und Mensch. In der Klasse *CSplitTreeINI* gibt es dafür die Funktion *SaveData2INI*. Der Funktion wird der Dateiname und der Entscheidungsbaum als Vector von *SplitTreeStruct* übergeben.

Für Präsentationszwecke und als zukünftige Speicherung der Bäume gibt es auch eine Sicherung als XML-Datei in der Klasse *CSplitTreeXML*. Um die Bäume visuell zu präsentieren wird das Programm „CASDraw“ aus der Abteilung „Grafische Ingenieursysteme“ der GFaI benutzt, welches Netzstrukturen automatisch anordnet.

4.3.2.2 Künstliches Neuronales Netz

Bei der Klassifikation mit einem NN wird der Backpropagation-Algorithmus verwendet (Abbildung 42). Als Netzstruktur werden Feed-Forward-Netze, Netze ohne Rückkopplung, benutzt.

Um ein Netz trainieren zu können, muss es erst „gebaut“ werden. Alle Input-, Hidden- und OutputLayer müssen erzeugt werden, ebenso die Verbindungen dazwischen. Das Neuronale Netz wird ein Objekt der Klasse *CNeuronalNet*. Über die Funktionen der Klasse können Neuronen und Verbindungen hinzugefügt und gelöscht werden.

Bei der Konstruktion ist zu beachten, dass Verbindungen eines Neurons immer eingehende Verbindungen sind. Ein Eingabeneuron hat also keine Verbindungen in den genutzten Feed-Forward-Netzen.

Jedes Eingabeneuron entspricht genau einem Merkmal, das bei der Datenvorbereitung gewählt wurde. Jedes Ausgabeneuron entspricht genau einer Klasse. Die Zahl der Hidden-Neuronen dagegen kann variiert werden und wird empirisch bestimmt. [Call03] schreibt zur Bestimmung der Anzahl von versteckten Neuronen folgendes:

> *„Eine Faustregel besteht darin, mit einer einzigen verborgenen Schicht zu beginnen, die 30-50% der Anzahl der Einheiten in der Eingabeschicht enthält.“*

Die Anzahl scheint gering, sollte aber genügen. Wenn es zu viele Neuronen sind, dann kann sich das Netz jedes einzelne Trainingsmuster mit zugehörigem Ausgabewert merken. Das Netz arbeitet als Speicher, es kann nicht mehr auf neue Daten richtig reagieren. Das Netz ist nicht generalisierungsfähig.

Nachdem das Netz generiert ist, kann es an *CLearnNeuronalNet* übergeben werden. Bevor es trainiert wird, sollten die Gewichte und der Schwellwert mit *initNet* initialisiert werden.

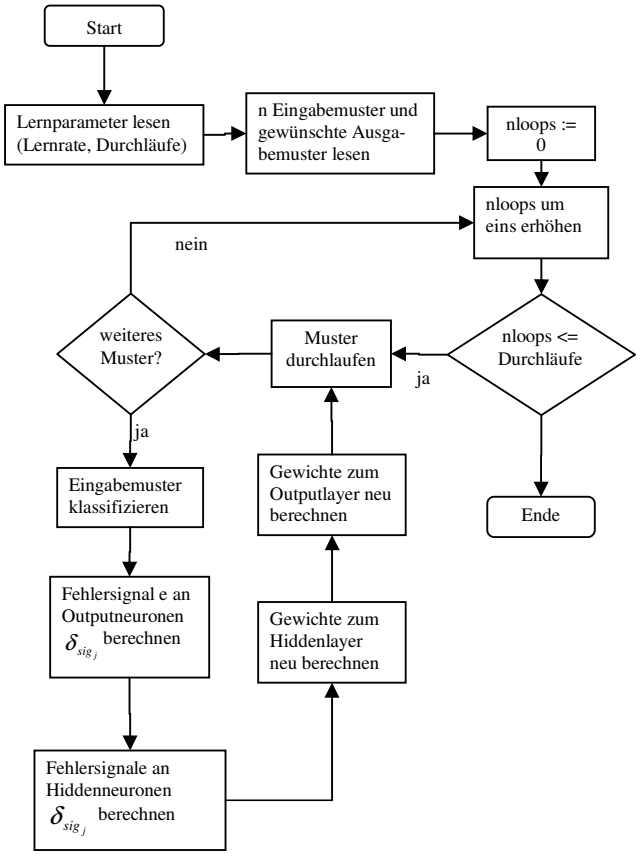

Abbildung 42: Backpropagation-Algorithmus

Mit *learnNet* beginnt der Algorithmus des Netzlernens. Der Funktion werden die Daten, die Lernrate η und die Anzahl der Durchläufe übergeben (Abbildung 42).

Die Umsetzung zum Aufbau eines Netzes funktioniert halbautomatisch. Die Eingabe- und Ausgabeneuronen werden automatisch erzeugt. Die versteckten Neuronen müssen vom Anwender festgelegt werden. Dabei gibt aber das Programm eine empfohlene Anzahl vor, die 50% der Anzahl der Eingabeneuronen entspricht. Anschließend muss der Benutzer festlegen, zwischen welchen Schichten Verbindungen erzeugt werden. Das Programm verbindet anschließend alle Neuronen der gewählten Schichten miteinander.

Das System ist nicht vollautomatisch, damit der Anwender bei einem schlecht trainierten Klassifikator eingreifen kann. Für eine reine manuelle Manipulation wird der Stuttgarter Neuronale Netz Simulator (SNNS) bzw. sein Java-Pendant JNNS empfohlen.

4.3.2.3 Stuttgarter Neuronale Netze Simulator

Das Stuttgarter Neuronale Netze Simulator [SNNS] ist ein Simulationsprogramm für Neuronale Netze. Es besteht seit dem Jahr 1989 und wird seitdem ständig weiter entwickelt. Für die Plattformunabhängigkeit wurde der SNNS von der Universität in Tübingen in Java programmiert. Der Java Neuronale Netz Simulator wurde zum Darstellen und Trainieren von Netzen für diese Arbeit verwendet.

Mit dem JNNS lassen sich Neuronale Netze auf einer grafischen Oberfläche erstellen. Diese Netze können sehr umfangreich aufgebaut sein. Der JNNS bietet zahlreiche Algorithmen für das Trainieren von Netzen unterschiedlichsten Aufbaus.

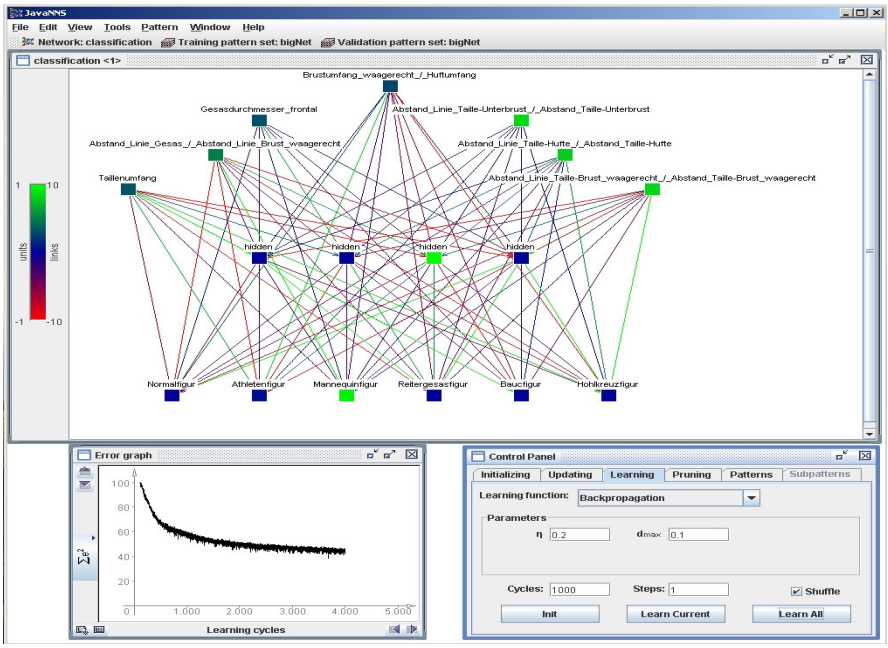

Abbildung 43: JNNS-Oberfläche

Damit der JNNS zum Darstellen, Manipulieren und Trainieren verwendet werden kann, wurde die Klasse *CJNNS2Interface* entwickelt. Mit der Funktion *saveNetData* kann ein Netz in ein für das JNNS auswertbares Format gespeichert werden. Mit *loadNetfromJNNS* kann das Netz wieder in das eigene Programm geladen werden.

4.3.3 Klassifizieren von Daten

4.3.3.1 Entscheidungsbaum

Um einen einzelnen Datensatz zu klassifizieren, muss nur ein Objekt der Klasse *CTreeClassify* erzeugt werden, dabei wird dem Konstruktor der Dateiname des Entscheidungsbaumes übergeben. Danach wird der Funktion *classifyData* der Dateiname der Messdaten übergeben und als Rückgabewert erhält man die *KlassenID*. Für den Klassennamen muss nur die Methode *getClassName* mit der ID aufgerufen werden.

Die Funktion zum Klassifizieren ist nur von sehr geringem Umfang. Der Baum wird von der Wurzel an durchlaufen bis zum Erreichen eines Blattes. Dabei wird immer der Ast verfolgt, der die *SplitRule*-Bedingung erfüllt (Abbildung 44).

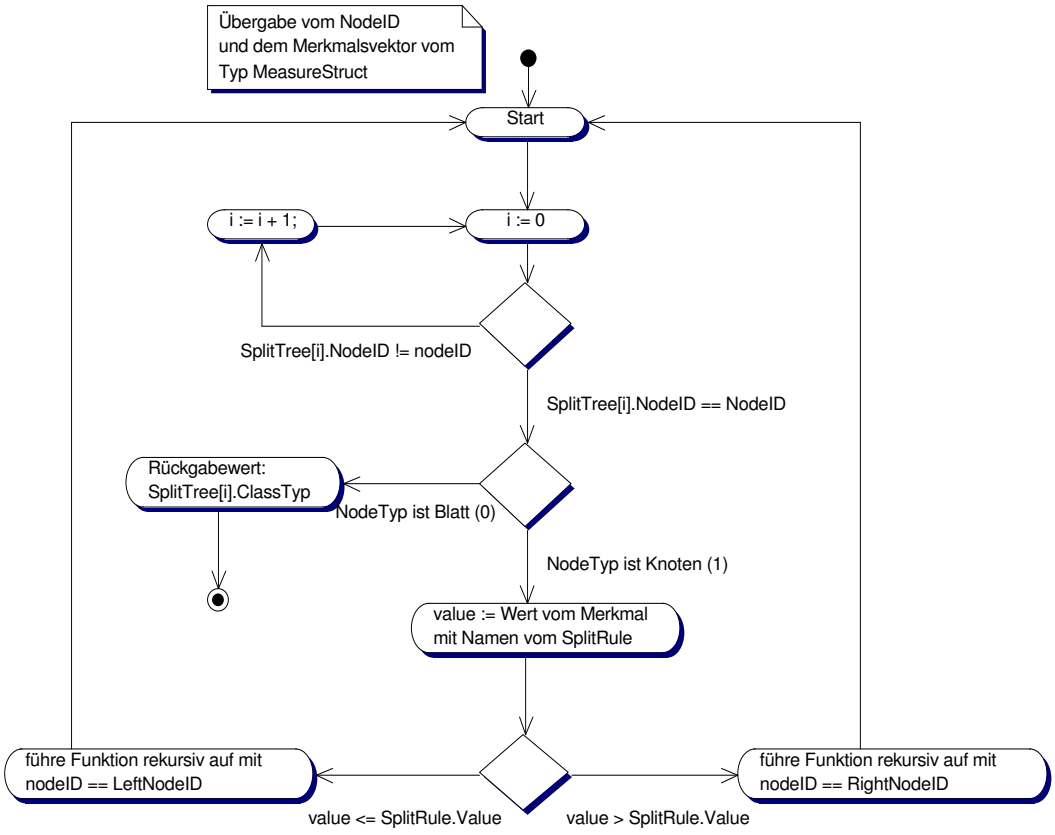

Abbildung 44: Aktivitätsdiagramm - Klassifizieren mit Entscheidungsbaum

Dieser Algorithmus ist auf alle binären Bäume anwendbar. Bei n-ären Bäumen muss aus der untersten Entscheidung eine Mehrfachentscheidung werden.

4.3.3.2 Neuronales Netz

Um einen einzelnen Datensatz zu klassifizieren, muss nur ein Objekt der Klasse *CNeuronNetClassify* erzeugt werden, dabei wird dem Konstruktor der Dateiname des Neuronalen Netz und der Name der Datei mit den zusätzlichen Skalierungsinformationen übergeben. In dem Prototyp werden derzeit nur Netze mit einer versteckten Schicht unterstützt. Danach wird die Funktion *classifyData* mit dem Dateinamen der Messdaten übergeben. Aus den Messdaten werden die für die Klassifikation wichtigen Merkmale in einen Merkmalsvektor geschrieben. Mit diesen Daten wird der in Abbildung 45 dargestellte Algorithmus durchlaufen. Als Rückgabewert erhält man die *KlassenID*. Für den Klassennamen muss nur die Methode *getClassName* mit der ID aufgerufen werden.

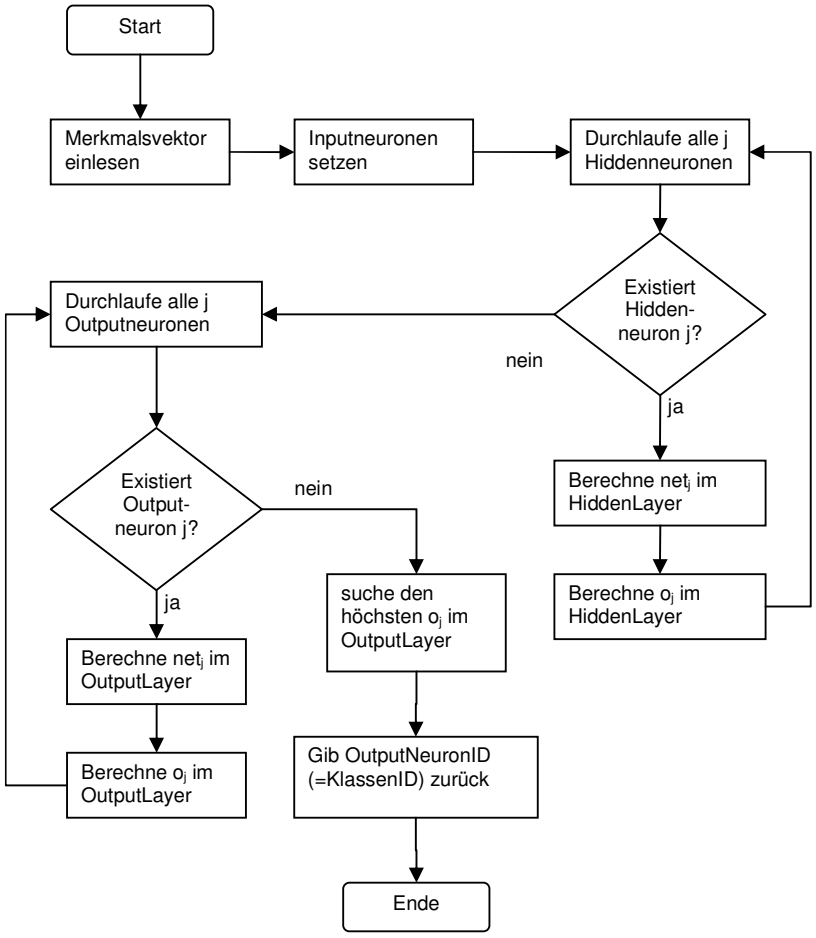

Abbildung 45: Klassifizieren mit Neuronalen Netzen

Zum Berechnen der Klassenzugehörigkeit müssen zuerst alle Inputneuronen mit den richtigen, konvertierten und skalierten Messwerten gesetzt werden. Dann werden alle Neuronen in der versteckten Schicht durchlaufen. Bei diesen Neuronen werden die Netzeingaben net_j und die Ausgaben o_j berechnet. Danach werden bei allen Outputneuronen die Netzeingaben net_j und Ausgabewerte o_j berechnet. Der aktivste Ausgabeneuron, bei dem o_j am höchsten ist, entspricht der Klassenzugehörigkeit.

4.4 Beschreibung der realisierten Lösung

Alle drei Prototypen haben eine ähnliche Oberfläche, die in drei Teile aufgeteilt ist. Im obersten Teil wird der Klassifikator erstellt und trainiert. Im mittleren Bereich kann der Klassifikator mit Testdaten auf seine Genauigkeit geprüft werden. Im untersten Bereich können einzelne Datensätze klassifiziert werden. Abbildung 46 zeigt die grafische Benutzerschnittstelle (GUI) für DecisionTree. Die Darstellungen der GUIs für die beiden anderen Prototypen befinden sich im Anhang A als Abbildung 50 und Abbildung 51.

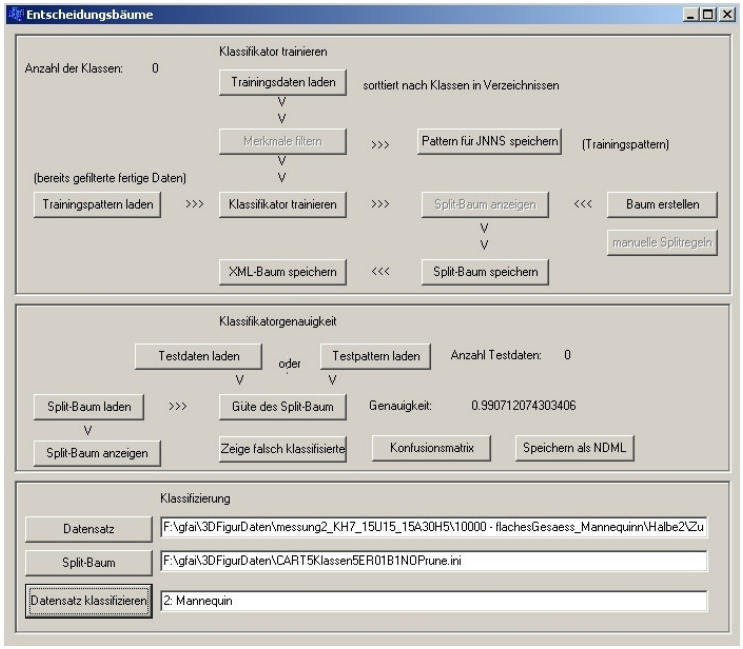

Abbildung 46: GUI – DecisionTrees

Für den Entwickler des Klassifikator sind dabei die beiden oberen Bereiche, Klassifikator trainieren und Klassifikatorgenauigkeit, von Bedeutung. Für den Endnutzer ist der untere Bereich Klassifizierung interessant, dort können einzelne Datensätze klassifiziert werden.

Die Entwicklung der GUI stand dabei nicht im Vordergrund. Auf grafische Ausgaben der Klassifizierung wurde verzichtet, zur Visualisierung wurde eine Schnittstelle zum 3D-Programm FinalSurfacer, eine Eigenentwicklung von 3DDV, implementiert.

Für die Beurteilung der entwickelten Klassifikatoren werden die Klassifizierungsgenauigkeit, die Konfusionsmatrix und die falsch klassifizierten Daten mit der *PersonID* angezeigt.

Für die richtige Reihenfolge in der Bedienung wurden im Dialogfenster Pfeile zwischen Buttons gesetzt, so dass deutlich wird, dass zum Beispiel kein Klassifikator ohne geladene Trainingsdaten trainiert werden kann.

4.5 Ergebnisse und Anwendung

4.5.1 Die Trainingsdaten

Für die Bewertung der einzelnen Algorithmen und der verschiedenen Klassifikatoren, die durch verschiedene Parameter entstehen, werden Trainingsmengen benötigt. Zunächst wurden künstliche 3D-Modelle verwendet, die den im Kapitel 3.1 beschriebenen Körpertypen entsprechen. Diese Figuren wurden nach den gleichen Regeln vermessen

wie die Personen in der Messkabine. Zusätzlich konnten auch Messungen durchgeführt werden, die bei den realen Personen noch nicht durchgeführt wurden.

Aber pro Klasse lag nur ein entsprechendes 3D-Modell vor. Deshalb wurden die künstlichen Daten vervielfacht. Dabei wurden Körperhöhe, Umfänge, Abstände zur Rückenabstandslinie und anderes nach dem Zufallsprinzip, aber innerhalb eines bestimmten Bereiches, variiert. Die veränderten Daten wurden noch mal geprüft und alle ungültigen wurden gelöscht.

So wurden 2 künstliche Trainingsmengen erzeugt, Trainingsmenge TR1 hat relativ eindeutige Grenzen zwischen den verschiedenen Körpertypen, so dass eine sehr hohe Klassifizierungsgenauigkeit erwartet wird (siehe Anhang Abbildung 52). Die zweite Trainingsmenge TR2 ist weniger genau in den Grenzregionen, die Körpertypen überlagern sich ein wenig, um so den realeren Daten zu gleichen, bei denen es auch nicht immer eindeutig ist, zu welcher Klasse eine Person zugehört (Abbildung 47).

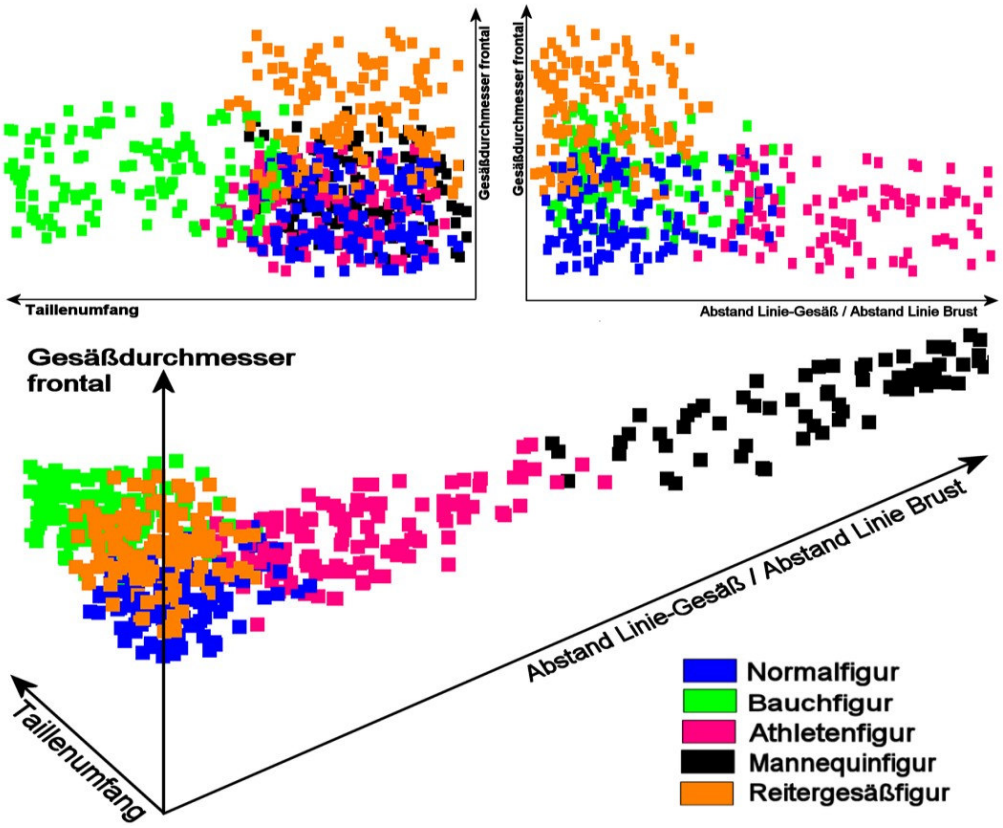

Abbildung 47: Trainingsmenge TR2

Bei dem Projekt „3D-Messkabine“ wurden schon über 600 Personen vermessen. Davon waren zum Zeitpunkt der Arbeiten ungefähr 400 Frauen, von denen über 150 aus den Anfangstagen der Messkabine sind und bei denen noch nicht alle für diese Klassifizierung notwendigen Maße gemessen worden waren. Bei einer weiteren Anzahl feh-

len ebenfalls noch wichtige Daten, so dass nur knapp weibliche 200 Messdaten für eine Klassifizierung zur Verfügung standen.

Für die Benutzung der Daten als Trainingsdaten müssen diese schon einer Körpertypklasse zugeordnet sein. Diese manuelle Zuordnung wurde erst nachträglich vom Autor für diese Arbeit vorgenommen und ist dadurch nicht als objektiv zu bezeichnen.

Ein weiterer Nachteil der Daten von den gemessenen Personen ist, dass nicht alle Körpertypklassen ausreichend vertreten sind, um eindeutige Grenzregionen zwischen den Klassen zu ermitteln. Aber wie erwartet, konnte die Mehrheit bei der Normalfigur-Klasse einsortiert werden. Der Bauchfigur konnten noch eine große Anzahl zugeordnet werden, aber die weiteren Typklassen haben maximal 10 Datensätze.

Die Hälfte der Originalmessdaten wurde als Trainingsmenge TR3 verwendet und die andere Hälfte als Testmenge TE3. Zusätzlich wurden die Mengen TR2 und TR3 vereinigt und bilden die TR4. So sollte untersucht werden, wie sich die wenigen Originaldaten auf die künstlichen Daten abbilden können, bzw. wie die Klassifikatoren auf Daten der Realität reagieren.

Zum Trainieren der Klassifikatoren werden pro Klasse ungefähr 64 Datensätze genutzt. Zum Testen der Klassifikationsgenauigkeit wird die gleiche Anzahl von Daten benutzt.

Verschiedene Kleidungsstücke benötigen eine unterschiedliche Anzahl von Körpertypen (siehe Kapitel 2.4), deshalb werden die Algorithmen mit 4, 5 und 6 Klassen trainiert und getestet. Für die 4-Klassen-Klassifizierung wurden die Typen: Normalfigur, Mannequinfigur, Bauchfigur und Reitergesäßfigur benutzt. Als 5. Klasse kam die Athletenfigur hinzu und als 6. die Hohlkreuzfigur. Es wären auch andere Kombinationen möglich gewesen. Dies ist abhängig von dem Bekleidungsstück, für welches die Personen in Körpertypen aufgeteilt werden sollen.

4.5.2 Auswertung der Verfahren

Zum Trainieren des kNN-Klassifikator wurde der Parameter k mit den Werten 3, 5, 7 und 10 verwendet. Dabei waren die Ergebnisse der Genauigkeit immer recht nah. Die Unterschiede lagen nur bis zu 4 Prozent auseinander. Die beiden besten Werte für k waren k=5 und k=7. Die Ergebnisse waren nahezu identisch.

Die Parameter für den EB-Klassifikator wurden auch variiert. Den größten Einfluss hat die Baumtiefe, die Anzahl der Ebenen. Bei 5 und 6 Klassen, die trainiert werden sollen, hat sich eine Baumtiefe von 7 Ebenen als der beste Wert ergeben. Bei weniger zu trainierenden Klassen genügen auch weniger Ebenen. Wenn mehr als 6 Klassen trainiert werden sollen, muss die beste Ebenenanzahl durch Vergleichen selbst ermittelt werden. Der Wert für die Reinheit eines Knotens wurde zwischen 0,1 und 0,01 getestet. Das derzeit umgesetzte Postpruning schneidet nur 2 gleich klassifizierende Blätter weg, so dass die Genauigkeit dadurch nicht beeinflusst wird, aber der Baum einwenig schrumpft.

Das vorwärts gerichtete Netz mit Backpropagation-Algorithmus eignet sich besonders gut für überwachtes Lernen [Call03]. Die Anzahl der Eingabe- und Ausgabeneuronen wird durch die Merkmale und die Klassenanzahl bestimmt. Variabel sind die Anzahl der versteckten Neuronen und die Verbindungen zwischen den Schichten. Getestet wurden Netze mit einer versteckten Schicht. Die Anzahl der versteckten Neuronen wurde bestimmt durch die Anzahl der Eingabeneuronen, sie wurde auf 50% festgelegt (siehe Kapitel 4.3.2.2). Die Neuronen wurden mit shortcut connections verbunden. Dies lieferte, im Vergleich zu einem Netz ohne direkte Verbindungen zwischen Eingabe- und Ausgabeschicht, die bessere Klassifizierung.

Das Trainieren eines Klassifikators kann je nach Algorithmus unterschiedlich viel Zeit in Anspruch nehmen. Beim kNN gibt es keine Trainierzeit, weil die Trainingsdaten schon den fertigen Klassifikator darstellen. Die EB benötigen bis zu 10 Sekunden, abhängig von der Anzahl der Klassen die ermittelt werden sollen und der Anzahl der Daten die zum Trainieren zur Verfügung stehen.

Das NN hat beim Trainieren den größten Zeitverbrauch. Besonders negativ ist der für diese Arbeit erzeugte Prototyp aufgefallen. Er benötigte für den Minimaltest mit 32 Datensätzen, 8 pro gewünschte Klasse und 2000 Lernzyklen eine gute Minute. Der erzeugte Klassifikator konnte die gleiche Klassifizierungsgenauigkeit erreichen wie ein durch den JNNS trainierter Klassifikator. Aber um den Klassifikator schneller und mit mehr Daten zu trainieren, wurde er mit dem JNNS trainiert. Auch mit dem JNNS war das Erlernen wesentlich langsamer als bei den Entscheidungsbäumen. Das Lernen dauerte 1 bis 2 Minuten. Die Klassifikatoren vom Prototypen und JNNS unterscheiden sich in den Gewichts- und Schwellwerten, weil sie mit zufälligen Werten initialisiert werden. Beide benutzen den gleichen Backpropagation-Algorithmus, aber bei der Programmierung des Algorithmus wurde beim JNNS eine bessere Performance erreicht. Der Gesamtfehler beim Trainieren erreichte bei beiden Programmen den ungefähr gleichen Wert.

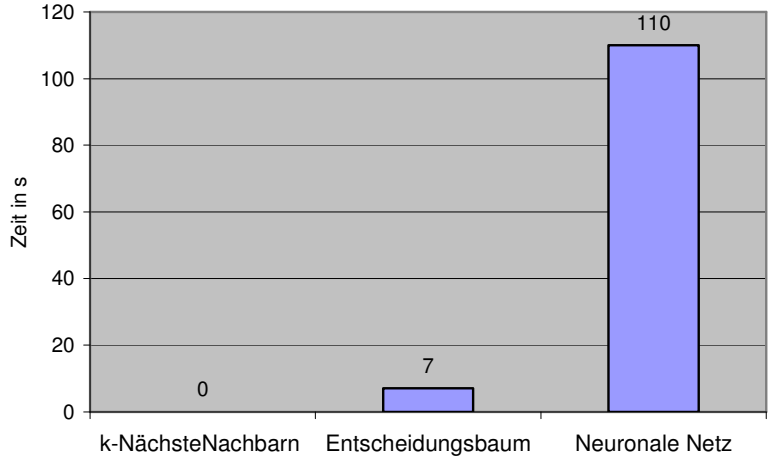

Abbildung 48: Diagramm Klassifikator-Lernzeit

In Abbildung 48 werden die verschiedenen Trainingszeiten miteinander verglichen. Der Klassifikator sollte dabei für 6 Klassen mit je 64 Trainingsdaten trainiert werden. Beim Neuronalen Netz wurde mit dem JNNS trainiert.

Das Klassifizieren ist bei allen 3 Algorithmen sehr schnell und kaum messbar. Bei einer wesentlich größeren Trainingsmenge wird der k-Nächste-Nachbarn-Algorithmus aber langsamer werden, weil er immer alle Daten mit der Distanzfunktion berechnen muss.

Unterschiede gab es bei der wichtigsten Eigenschaft der Klassifikatoren, der Klassifikationsgenauigkeit G. Am schlechtesten schnitt der k-Nächste-Nachbarn ab. Je mehr Klassen und Merkmale trainiert werden mussten, umso schlechter waren die Klassifizierungen der Testdaten.

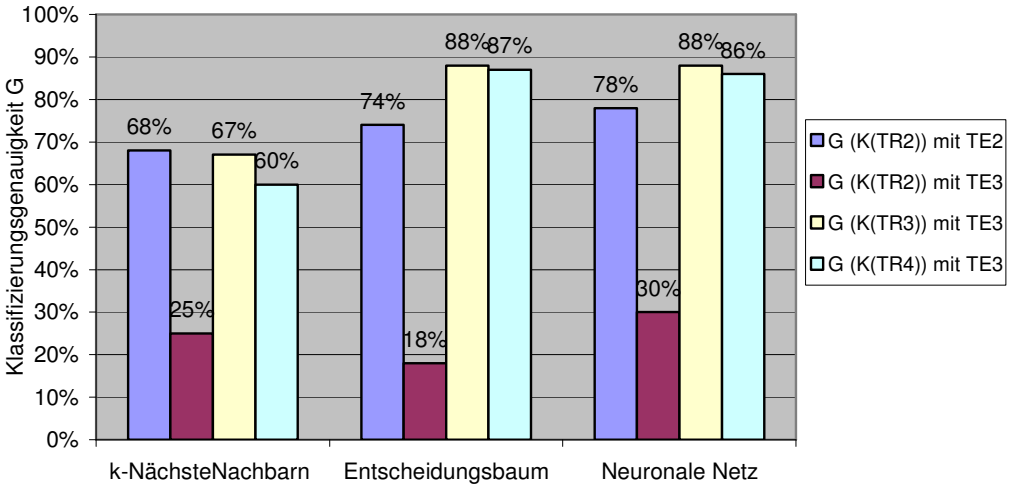

Abbildung 49: Diagramm Klassifikatorgenauigkeit

Wenn der Klassifikator mit der künstlichen Menge TR2 trainiert und mit der natürlichen Menge TE3 getestet wurde, sank die Genauigkeit rapide ab (18% - 30%). Dafür kommen 2 Gründe in Frage: 1. Die realen Daten wurden vorher nicht richtig geordnet, 2. die Relationen der Maße von künstlichen und realen Figuren sind zu unterschiedlich, um sie mit dem gleichen Klassifikator zu klassifizieren.

Für die spätere Nutzung ist es wünschenswert, dreidimensionale Figuren zur Verfügung zu haben, deren Proportionen und dadurch auch Maße so veränderbar sind, dass der Schneider vollen Einfluss auf die unterschiedlichen Körpertypen nehmen kann. Für unterschiedliche Bekleidungsstücke können die Definitionen der Körpertypen variieren, dann sollte man auch die Trainingsdaten der Körpertypen verändern können, visuell aber auch die Maße. Dafür müssten die aktuellen virtuellen Figuren überarbeitet werden. Um nur reale Figuren zu nutzen, müsste immer eine rausreichend große Menge von Messdaten mitgeliefert werden. Der Designer müsste dann die Daten, vor allem die Grenzregionen, aufwendig neu klassifizieren.

Die Genauigkeit des Entscheidungsbaum-Klassifikators ist mit TR2 etwas geringer als die des NN. Das liegt an den sehr ungenauen Grenzen von TR2. Je klarer die Grenzen sind, umso besser ist der Klassifikator. Das neuronale Netz kann trotz nicht ganz eindeutiger Trainingsdaten besser generalisieren.

Das Speichern der trainierten Klassifikatoren ist nicht von großer Bedeutung. Die Dateigrößen bewegen sich im Bereich von 1 kB bis 20 kB, je nach Anzahl der Klassen und Merkmalen.

Ein interessanter Unterschied ist die Interpretierbarkeit der Modelle. Soll der Anwender eine Klassifizierung ohne große Rechnung nachvollziehen können oder nicht? Beim kNN ist dies nur bei bis zu 3 Merkmalen grafisch möglich. Beim NN ist es ohne sehr großen Rechenaufwand nicht nachvollziehbar, wie es zu einer Klassifizierung einer Person kommt. Beim EB ist es dagegen relativ leicht, die Klassifizierung selbst durchzuführen, wenn der Baum sichtbar ist.

All die aufgezählten, unterschiedlichen Eigenschaften der Klassifizierungen lassen sich in einer Tabelle zusammenfassen:

Tabelle 3: Vergleich der Klassifizierungen

	k-Nächste-Nachbarn	Entscheidungsbaum	Neuronales Netz
Klassifikations-Genauigkeit	niedrigste Klassifikations-genauigkeit	gute Klassifikationsge-nauigkeit	beste Klassifikationsge-nauigkeit
Trainingszeit	Nicht vorhanden	sehr gute Trainingszeit; im Sekundenbereich	langsames Trainieren; mit eigenem Algorithmus zu langsam; mit JNNS im 1-2 min Bereich
Interpretierbarkeit	bei n-Dimension nicht unbedingt nachvollzieh-bar	Klassifizierung nachvoll-ziehbar	Klassifizierung nicht nachvollziehbar
Klassifizierungszeit	gleich schnell	gleich schnell	gleich schnell
Kompaktheit des Modells	je mehr Trainingsdaten desto größer der Klassifikator	recht klein, abhängig vom Baum	recht klein, Neuronen mit gewichteten Verbindun-gen

In der Tabelle 3 sind die Vor- und Nachteile jeder Klassifizierung gegenübergestellt. Wichtig sind die Prioritäten eines Nutzers an den Klassifikator, wobei die Klassifizierungsgenauigkeit den höchsten Stellenwert einnehmen sollte. Das Neuronale Netz hat insgesamt die beste Klassifizierungsgenauigkeit erreicht. Sind die Trainingszeit und Interpretierbarkeit des Klassifikators nicht von Bedeutung, kann das NN benutzt werden.

Der EB ist bei der Genauigkeit nur minimal schlechter. Deshalb bietet er sich auch als ein guter Klassifikator an. Zusätzlich hat er eine wesentlich schnellere Trainingszeit und bessere Interpretierbarkeit. Der kNN kann für die Klassifizierung von Körpertypen nicht empfohlen werden, da die Klassifizierungsgenauigkeit im Vergleich zu den anderen Klassifikatoren zu gering ausgefallen ist.

Zur Klassifizierung von Körpertypen können dem Nutzer die trainierten Klassifikatoren vom Neuronalen Netz und vom Entscheidungsbaum empfohlen werden. Aber beim Trainieren sollten jeweils mehrere Klassifikatoren mit anderen Lernparametern trainiert werden. Erst durch das Vergleichen der einzelnen Genauigkeiten sollte dann die endgültige Wahl gefällt werden.

5 Zusammenfassung

Als Ergebnis dieser Arbeit wurde eine automatische Klassifizierung von Körpertypen entwickelt. Dafür wurden verschiedene Wege der Klassifizierung untersucht und bewertet. Das Neuronale Netz hat die höchste Genauigkeit beim Klassifizieren erreicht. Die Klassifizierung mit einem Entscheidungsbaum konnte auch überzeugen, dazu war die Lernzeit des Klassifikators wesentlich schneller.

Für eine Klassifizierung steht die Einordnung von Personen in Körpertypen an oberster Stelle, aber ohne einen gut trainierten Klassifikator wird keine Klassifizierung erfolgreich. So wurden die Eigenschaften einer guten Trainingsmenge beschrieben und wie mit dieser ein Klassifikator gebildet wird.

Wichtig sind vor allem aussagekräftige Trainings- und Testdaten, die die Wirklichkeit auch sehr gut repräsentieren können. Am besten eignen sich dazu reale Daten in ausreichender Menge. Leider standen während der Erstellung dieser Arbeit noch nicht genügend Daten zur Verfügung, die die verschiedenen Klassen ausreichend abdeckten. Ein Klassifikator, der nur auf Daten der virtuellen 3D-Modelle basiert, konnte nicht auf reale Daten angewendet werden. Dafür müssen die zugrunde gelegten Avatare den realen Daten besser nach empfunden werden.

Es wurden die Körpertypklassen vorgestellt, die bei einer individuellen Schnittkonstruktion mit beachtet werden müssen. Die dafür notwendigen Maße als Merkmale für die Klassifizierung wurden ermittelt. Für verschiedene Kleidungsstücke werden unterschiedliche Typklassen benötigt, deshalb wurde das System auf keine festgelegten Klassen und Merkmale programmiert. Diese können je nach Wunsch selbst angelegt werden. So können die Funktionen auch für die Herrenbekleidung benutzt werden, wenn vorher die Typklassen und die dazugehörigen Maße definiert werden.

In Zukunft wäre es denkbar, dass durch die Klassifikation unterschieden werden kann, ob die gemessene Person männlich oder weiblich ist. Dafür müssten aus den gemessenen Körpermaßen, die für die Klassifizierung wesentlichen Maße ermittelt werden. „Unterbrustumfang“ und „Brustumfang waagerecht“ würden dafür unter anderem in Betracht kommen. Aber besonders bei dieser Klassifizierung sind die Grenzregionen sehr entscheidend, denn eine Falschklassifizierung beim Geschlecht wäre wohl extrem kundenunfreundlich.

Das entwickelte Verfahren kann als prototypische Lösung betrachtet werden. Es ergeben sich weitere Möglichkeiten, die Umsetzung zu verbessern. Das Postpruning des entstandenen Baumes wurde nur ansatzweise umgesetzt und bietet daher noch Möglichkeiten für eine Verbesserung. Des Weiteren wurde nur der binäre Baumbildungsalgorithmus CART untersucht, die Verfahren C4.5 und ID3 unterstützen auch Multisplitbäume.

Anhang A: Diagramme und Tabellen

A1 Benutzeroberflächen

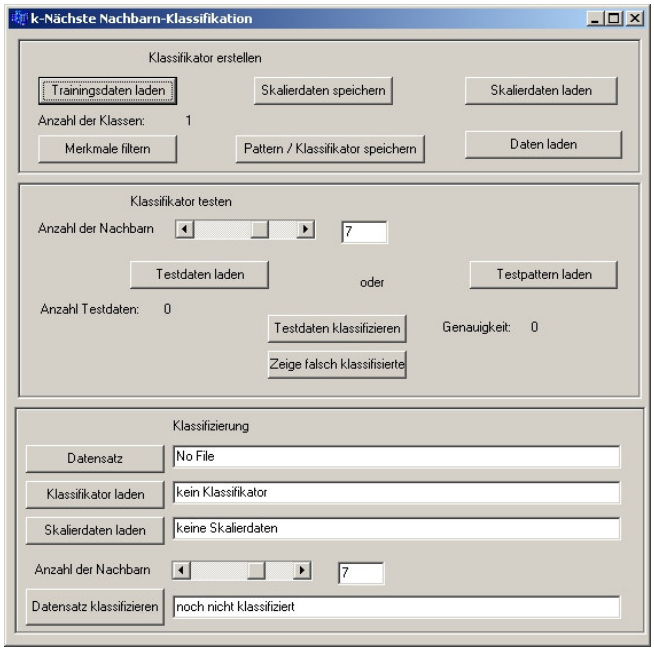

Abbildung 50: GUI – kNearestNeighbor

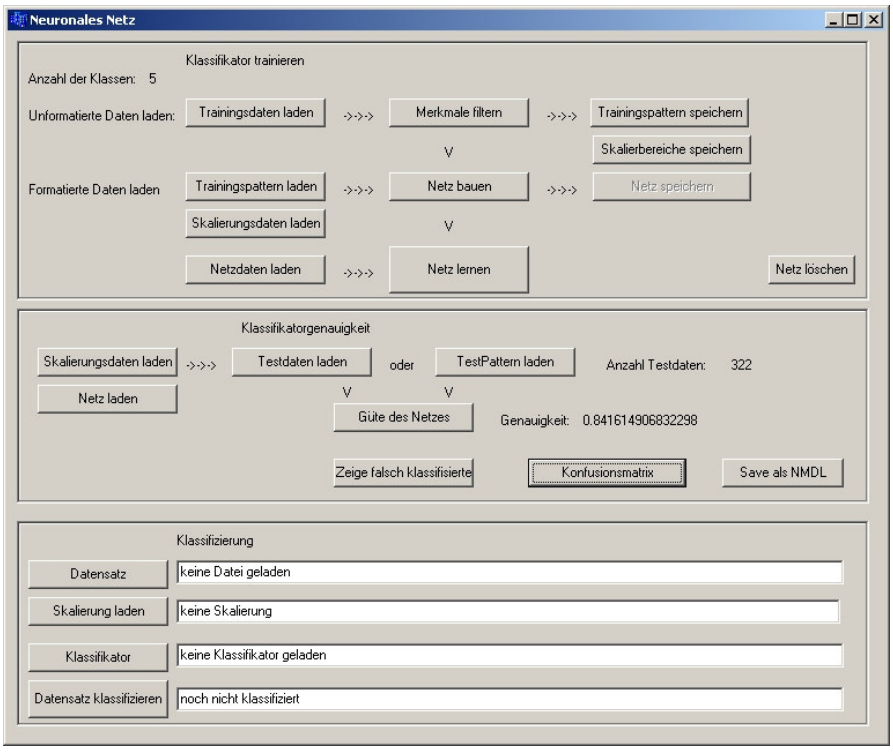

Abbildung 51: GUI – NeuronalClassify

A2 Trainingsmenge TR1

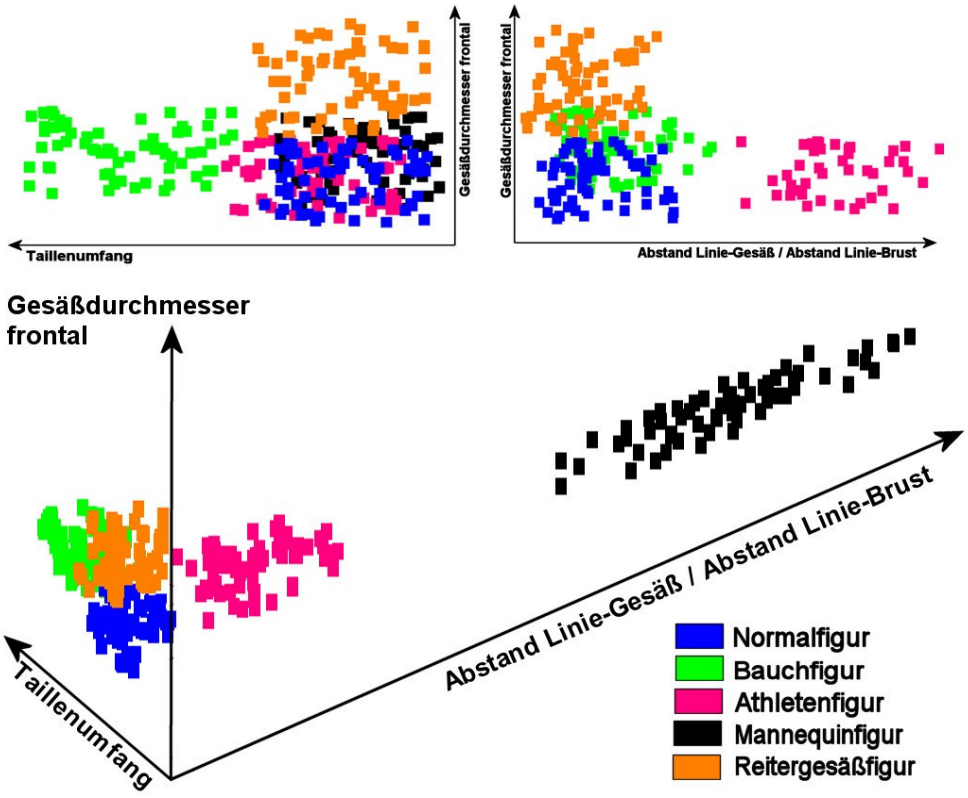

Abbildung 52: Trainingsmenge TR1

A3 Beispielprotokoll

Beispiel: Klassifikator erstellen und testen

Es wird mit der TR2 ein Klassifikator generiert. Als Parameter für den EB werden die Einstellungen von Tabelle 4 verwendet. Es sollen die Klassen Normalfigur, Mannequinfigur, Bauchfigur, Reitergesäßfigur und Athletenfigur trainiert werden. Als Testmenge wird TE2 verwendet. Die Abbildungen Abbildung 47, Abbildung 53 und Abbildung 54 sind die Darstellungen der jeweiligen Klassifikatoren.

Tabelle 4: Entscheidungsbaum - Lernparameter

Maximale Baumtiefe	7
Minimale Reinheit	0,01
Maximale Anzahl von Objekten bei Unreinheit	3
Anzahl Testmengen	10

Tabelle 5: Neuronales Netz - Lernparameter

Initialisierungsbereich für die Gewichte	-0,5 – 0,5
Lernrate η	0,5
Anzahl Durchläufe	20000

Tabelle 6: Klassifikator - Überblick

	k-Nächste-Nachbarn	**Entscheidungs-baum**	**Neuronales Netz**
Trainingszeit	0,0s	2,77s	60,0s
Klassifizierungsgenauigkeit	0,83	0,88	0,89

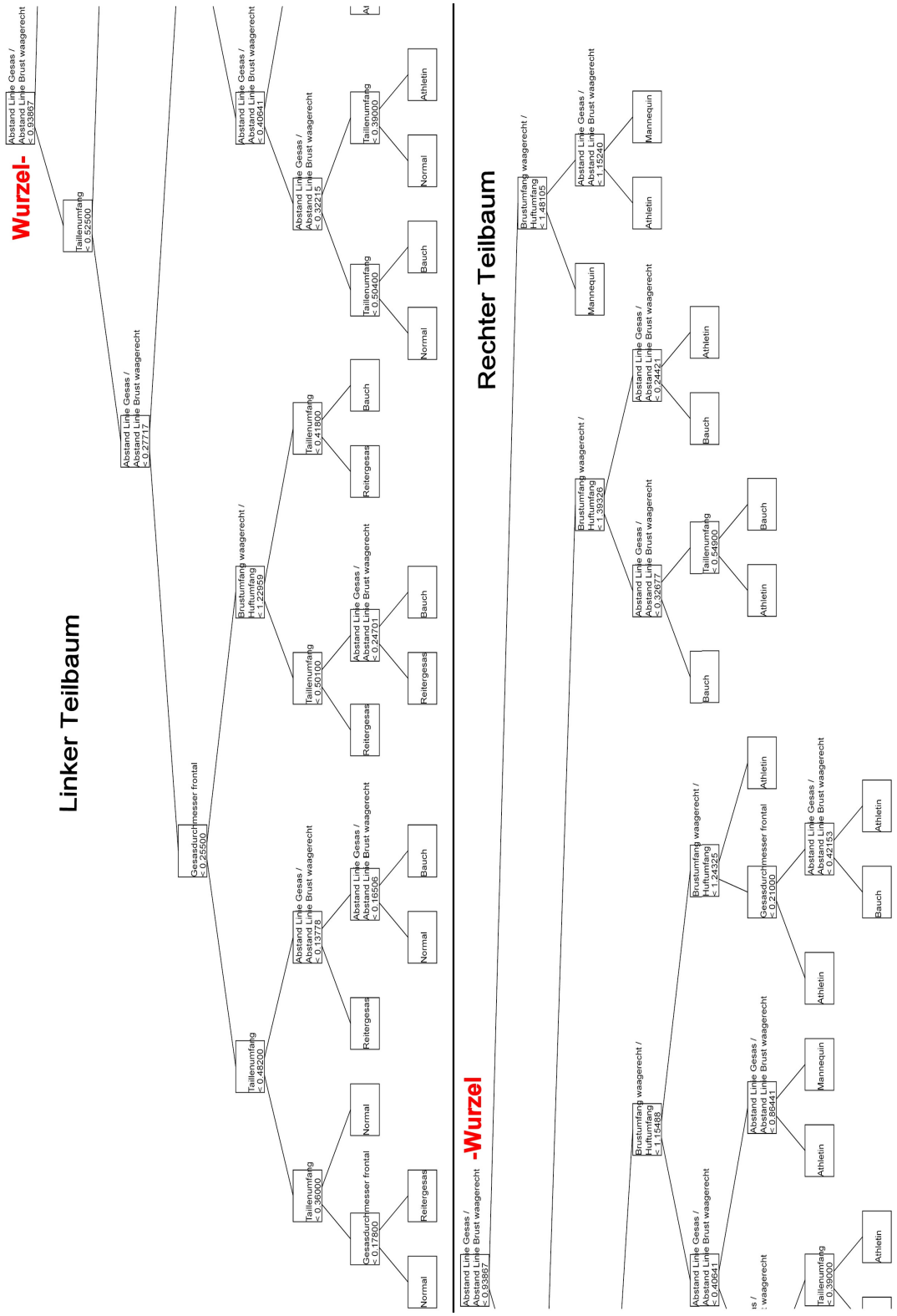

Abbildung 53: Entscheidungsbaum-Klassifikator

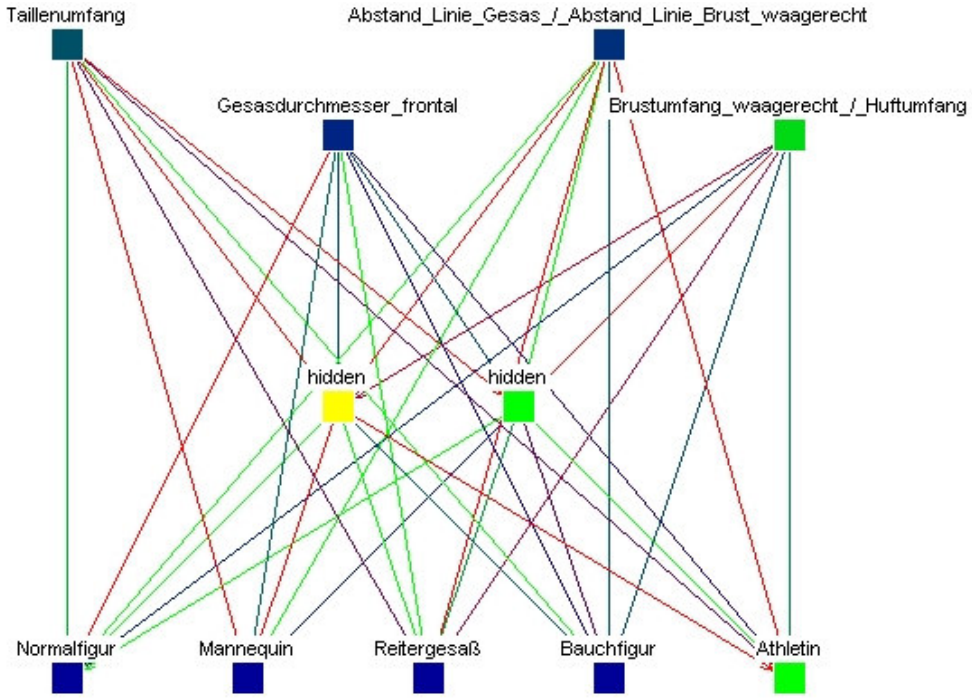

Abbildung 54: Neuronales Netz-Klassifikator

Abbildung 55 zeigt die Testmenge die durch den EB klassifiziert wurde. Die klassifizierten Testmengen von kNN und NN sehen sehr ähnlich aus, wobei beim kNN mehr falsch klassifizierte Objekte vorkommen. Die falsch klassifizierten Objekte befinden sich hauptsächlich in den Grenzregionen zwischen den verschiedenen Klassen.

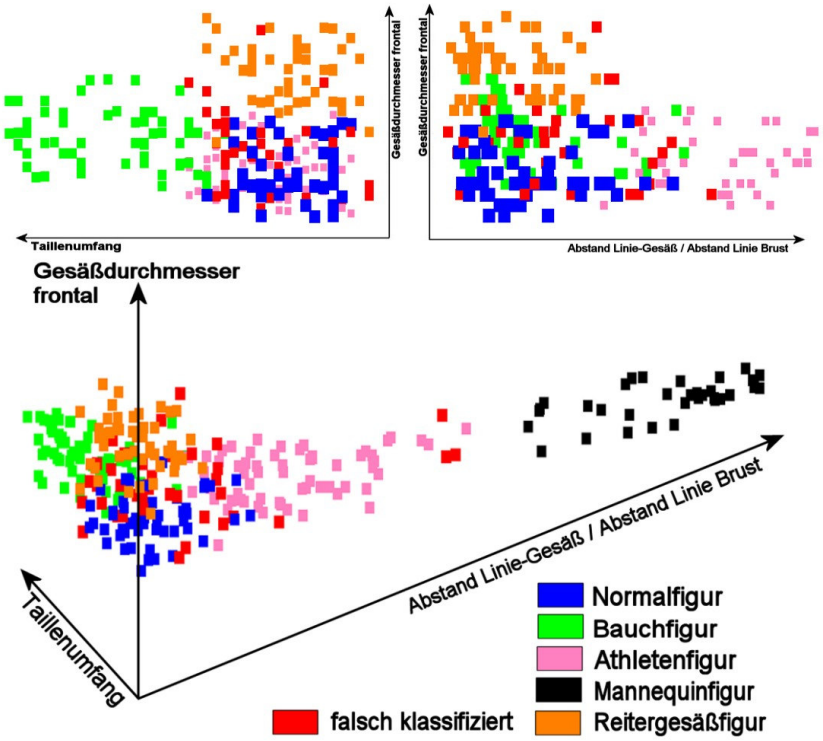

Abbildung 55: klassifizierte Objekte

Es folgen nun die Konfusionsmatrizen für den EB- und NN-Klassifikator (Tabellen Tabelle 7 und Tabelle 8). Wie in Abbildung 11 sind in der 1. Spalte die tatsächlichen Klassen der Objekte aufgezählt und in der 1. Zeile die vom Klassifikator zugeordneten Klassen. Eine Auswertung aller Tests ist in Kapitel 4.5.2 „Auswertung der Verfahren“ beschrieben.

Tabelle 7: Konfusionsmatrix Entscheidungsbaum

	Normalfigur	Athletin	Mannequin	Reitergesäß	Bauchfigur
Normalfigur	51	4	0	4	5
Athletin	3	60	1	0	0
Mannequin	0	2	64	0	0
Reitergesäß	11	0	0	49	4
Bauchfigur	2	1	0	3	58

Tabelle 8: Konfusionsmatrix Neuronales Netz

	Normalfigur	Athletin	Mannequin	Reitergesäß	Bauchfigur
Normalfigur	57	1	0	4	2
Athletin	5	59	0	0	0
Mannequin	0	2	64	0	0
Reitergesäß	10	0	0	53	1
Bauchfigur	5	3	0	2	54

Anhang B: Glossar

Abnäher: Abnäher sind in der Schnittkonstruktion Einschnitte im Stoff, die zusammengenäht werden.

Aktivitätsdiagramm: Mit einem Aktivitätsdiagramm wird das Verhalten komplexer Anwendungsfälle analysiert und die Interaktion zwischen ihnen dargestellt.

Anthropometrie: Anthropometrie ist die Lehre von den Maßen des menschlichen Körpers.

Data Mining: Data Mining ist das systematische Entdecken und Extrahieren unbekannter, nicht trivialer und wichtiger Informationen aus großen Mengen von Daten

Gradierung: Gradierung bedeutet in der Schnittkonstruktion die lineare Veränderung der Schnittparameter, z.B. für eine Größenanpassung.

Metrik: Eine Metrik beschreibt in der Mathematik eine Funktion, die den Abstand zweier Punkte eines Raums einen reellen Wert zuordnet.

Normieren: Beim Normieren werden alle Werte eines Parameters in ein festgelegten Bereich skaliert, z.B. zwischen 0,1 und 0,9.

Rapid Prototyping: Rapid Prototyping bezeichnet Verfahren zur Herstellung eines Prototypen eines Produktes mit verhältnismäßig geringem Aufwand.

Sigmoide Funktionen: Sigmoide Funktionen bezeichnen in der Mathematik Funktionen, die in ihrer grafischen Darstellung dem Buchstaben „S“ ähnlich sind.

Use Case Diagramm: Ein Use Case beschreibt die Anwendungsfälle einer Anwendung, die beteiligten Akteure sowie die Beziehungen zwischen diesen beiden Mengen.

Anhang C: CD-ROM

Inhalt der CD

1. Quellcodes der Prototypen
2. Prototypen als ausführbare Dateien
3. Trainings- und Testmengen
4. Diplomarbeit im PDF-Format

Abkürzungsverzeichnis

3DDV	3D Datenverarbeitung
EB	Entscheidungsbaum-Algorithmus
CART	Classification and Regression Tree
GFaI	Gesellschaft zur Förderung angewandter Informatik
GUI	Graphical User Interface
JNNS	Java Neuronaler Netz Simulator
KI	Künstliche Intelligenz
kNN	k-Nächste-Nachbarn-Algorithmus
NN	künstliches Neuronales Netz
SNNS	Stuttgarter Neuronale Netze Simulator
SVM	Support Vector Machines

Literaturverzeichnis

[Beck86] **Beck, U.:**
Risikogesellschaft – auf dem Weg in eine andere Modern
Frankfurt, 1986

[Brause91] **Brause, R.:**
Neuronale Netze – Eine Einführung in die Neuroinformatik
B.G. Teubner, Stuttgart, 1991,S. 3f

[Call03] **Callan, R.:**
Neuronale Netze im Klartext
Pearson Studium, Paderborn 2003, S.20-32, 50-53,155-160

[FKPT97] **Fahrmeir, L.; Künstler, R.; et al.:**
Statistik - der Weg zur Datenanalyse
Springer, Berlin, 1997

[Four94] **Fournier, G.:**
Informationstechnologien in Wirtschaft und Gesellschaft
Berlin, 1994

[Heuw02] **Heuwold, N.; Paul, L.:**
Aktuelle Ansätze und Lösungen zur berührungslosen Körpermaßerfassung unter dem Aspekt der Realisierung von automatischen Messkabinen
5. Workshop GFaI Berlin, Tagungsband 3D-Nord-Ost 2002

[Hoff91] **Hoffmann, N.:**
Simulation neuronaler Netze – Grundlagen, Modelle, Programme
Vieweg, Braunschweig, 1991

[Kirch01] **Kirchdörfer**, **E.:**
Bekleidungstechnische Schriftenreihe Band 144: Grundlagen für die 3-D-Konstruktion und Passformsimulation körperferner Kleidungsstücke
Köln, 1992, S. 29

[Lämm01] **Lämmel, U.; Cleve, J.:**
Künstliche Intelligenz: Lehr- und Übungsbuch
Fachbuchverlag Leipzig, 2001, S. 171-195

[Lenz97] **Lenze, B.:**
Einführung in die Mathematik neuronaler Netze
Logos-Verlag Berlin, 1997

[Paul04] **Paul, L.;Wilde, G:**
Bausteine der automatisierten industriellen Maßkonfektion
Melliand-Bekleidung, 1-2/2004, S96-98

[Schnitt] **Schnitt-Technik:**
Damen Rundschau, Rundschau-Verlag
Ausgabe 7/2001, S. 90-98
Ausgabe 12/2001, S. 23-30

[Scit89] **Scitovsky, T.:**
Psychologie des Wohlstands: Die Bedürfnisse des Menschen und der Bedarf der Verbraucher
Frankfurt, 1989

[Seid01] **Seidl, A.; Mecheels, S.; et al.:**
Zukunft Maßkonfektion: Technik, Markt und Management
Frankfurt am Main, 2001, S.17-35, 92-97

[Stieg92] **Stiegler, M.:**
Schnittkonstruktionen für Kleider und Blusen
Hrsgb: Deutsche Bekleidungs-Akademie München
Rundschau-Verlag Otto G. Königer GmbH & Co. München 1992, S.235-254

[Textil04] **Kern, J.:**
Umsatz nach Maß
TextilWirtschaft 2004,Frankfurt am Main, Heft 44, S. 32-35

Internetquellen

[BodyFit] **BodyFit 3D:**
www.bodyfit3d.de
(Datum des Zugriffs: 10. Oktober 2004)

[Derb00] **Derbsch, H.:**
Data Mining im Marketing – Theorie zu Entscheidungsbäumen
www.ku-eichstaett.de/Fakultaeten/WWF/ Lehrstueh-le/WI/Lehre/dm_v/Sections/content/DM%207.pdf
(Datum des Zugriffs: 25. September 2004)

[Ganter] **Ganter, H.-D.; Lieb, M. G.;et al.:**
Krisenmanagement in der Bekleidungsindustrie
http://intra.fh-heilbronn.de/IAF/Presse/Berichte/Bekleidung.html
(Datum des Zugriffs: 8. Oktober 2004)

[Human] **Human Solutions GmbH:**
http://www.human-solutions.de
(Datum des Zugriffs: 29. Oktober 2004)

[Köst03] **Köster, F.:**
Data Warehousing and Knowledge Discovery in Databases – Klassifikation IV
www-is.informatik.uni-oldenburg.de/~koester/ Vorlesung/DWH-und-KDD--VL-15---Klassifikation-IV.pdf
(Datum des Zugriffs: 13. September 2004)

[JNNS] **Java Neural Network Simulator:**
http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome.html
(Datum des Zugriffs: 8. Oktober 2004)

[Pill00] **Piller, F. T.:**
Mass Customization News – Nr. 12
http://www.mass-customization.de/news/news00_12.pdf
München, 2000
(Datum des Zugriffs: 13. Juni 2004)

[Pretz03] **Pretzer, M.:**
www.diko-project.de/dokumente/ ausarbeitungen/**pretzer**.pdf
(Datum des Zugriffs: 10. Mai 2004)

[SNNS] **Stuttgart Neural Network Simulator:**
http://www-ra.informatik.uni-tuebingen.de/SNNS/
(Datum des Zugriffs: 8. Oktober 2004)

[SYMCAD] **SYMCAD:**
http://www.symcad.com/de/technology2.htm
(Datum des Zugriffs: 20. Oktober 2004)

Stichwortverzeichnis

Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Ort, Datum

Unterschrift