Analyse dynamischer Lernregeln für Case-Based Learning Systeme

Ernst-Georg Haffner

Diplomarbeit

Analyse dynamischer Lernregeln für Case-Based Learning Systeme

ERNST-GEORG HAFFNER

10. Februar 1993

AG Künstliche Intelligenz – Expertensysteme

Prof. Dr. Michael M. Richter

Sonderforschungsbereich 314
„Künstliche Intelligenz – Wissensbasierte Systeme“
Projekt X9

Betreuung :

Dipl.-Inform. Stefan Weß

Erklärung

Ich versichere, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kaiserslautern, den 10. Februar 1993

Zusammenfassung

Die vorliegende Arbeit konzentriert sich auf Analysen unterschiedlicher Lernstrategien für CBL-Systeme anhand einer universellen Testumgebung mit variablen Fallbasen.

Keine der untersuchten dynamischen Lernregeln und keine feste Belegung der globalen Konstanten im Ähnlichkeitsmaß besitzt im statistischen Mittel signifikante Vorzüge. Dagegen zeigen sich Abhängigkeiten des Lernerfolgs von bestimmten Merkmalen der Fallbasis.

Deswegen wird als Synthese ein *auto-adaptives Lernschema* vorgeschlagen, das die Eigenheiten verschiedener Fallbasen berücksichtigt und durch die Wahl spezifischer Lernstrategien ein deutlich verbessertes Ergebnis zu erzielen vermag.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Symbol	Beschreibung	Seite
$\mathcal{P}$	Problem (2.1)	6
$\mathcal{N}$	Anzahl der Merkmale (2.1)	6
$\perp$	unbekannter / beliebiger Attributwert (Def. 2.1)	6
$\mathcal{B}$	Problembasis (2.1)	6
$\mathcal{L}$	Menge aller (Klassifikations-)Lösungen	7
$\mathcal{L}_\mathcal{P}$	Lösungen des Problems $\mathcal{P}$	7
$\vec{l}$	Lösungsvektor (2.2)	7
$\mathcal{R}$	Relevanzmatrix	18
ω_{ij}	Element der Relevanzmatrix (Zeile i, Spalte j)	18
ω_{ij}^A	ω_{ij} *vor* der Relevanz-Änderung (altes Gewicht)	32
ω_{ij}^N	ω_{ij} *nach* der Relevanz-Änderung (neues Gewicht)	32
ζ	Attribut-Ähnlichkeits-Funktion (4.4)	19
$a_\mathcal{P}^i$	i-tes Merkmal des Problems $\mathcal{P}$ (kurz $a_\mathcal{P}$) (4.4)	19
a_{Fall}^i	i-tes Merkmal des Falles (kurz a_{Fall}) (4.4)	19
a_{Klasse}^i	i-tes Merkmal der Klasse (kurz a_{Klasse}) (4.5)	19
θ	Schwellwert erfüllter Attribute (4.5)	19
$\mathcal{C}_k^j$	Fall $\mathcal{C}_k$ mit Relevanzbezug zur Klasse $\mathcal{K}_j$ (kurz $\mathcal{C}_k$)	20
$\mathcal{E}_{\mathcal{P},\mathcal{C}_k}$	Menge erfüllter Merkmale von $\mathcal{P}$ und $\mathcal{C}_k$ (kurz $\mathcal{E}$)	20
$\mathcal{W}_{\mathcal{P},\mathcal{C}_k}$	Menge widersprüchlicher Merkmale von $\mathcal{P}$ und $\mathcal{C}_k$ (kurz $\mathcal{W}$)	20
$\mathcal{U}_{\mathcal{P},\mathcal{C}_k}$	Menge unbekannter Merkmale von $\mathcal{P}$ und $\mathcal{C}_k$ (kurz $\mathcal{U}$)	20
$\mathcal{Z}_{\mathcal{P},\mathcal{C}_k}$	Menge der zusätzlichen Merkmale von $\mathcal{P}$ und $\mathcal{C}_k$ (kurz $\mathcal{Z}$)	20
E	gewichteter Ähnlichkeitswert der Attribute aus $\mathcal{E}$ (4.9)	21
W	gewichteter Ähnlichkeitswert der Attribute aus $\mathcal{W}$ (4.9)	21
U	gewichteter Ähnlichkeitswert der Attribute aus $\mathcal{U}$ (4.9)	21
Z	gewichteter Ähnlichkeitswert der Attribute aus $\mathcal{Z}$ (4.9)	21
δ_j	Schwellwert der Klasse $\mathcal{K}_j$	30
Δ	Überschreitung der Ähnlichkeit ohne gemeinesame Diagnose	30
$\Delta_\mathcal{E}$	Maß für Änderung an erfüllten Attributwerten (5.2)	31
$\Delta_\mathcal{W}$	Maß für Änderung an widersprüchlichen Attributwerten (5.2)	31
$\Delta_\mathcal{U}$	Maß für Änderung an unbekannten Attributwerten (5.2)	31
E_{Neu}	Kurzschreibweise für: $E - \Delta_\mathcal{E}$	31
W_{Neu}	Kurzschreibweise für: $W + \Delta_\mathcal{W}$	31
U_{Neu}	Kurzschreibweise für: $U + \Delta_\mathcal{U}$	31
f, f_S	f Verteilungsfunktion, f_S erzeugende Funktion von f	32
$\chi_\mathcal{E}$	Kurzschreibweise für $\sum_{a \in \mathcal{E}} \omega_{ij}^A$ (5.17)	34
$\chi_\mathcal{W}$	Kurzschreibweise für $\sum_{a \in \mathcal{W}} \omega_{ij}^A$ (5.17)	34

Teil I

Grundlagen

Kapitel 1

Einleitung und Überblick

1.1 Einführung

Unter der Vielzahl von möglichen Modellierungen menschlichen Lernens für den automatischen Einsatz auf Computern tritt die Bedeutung des Erfahrungswissens immer stärker in den Vordergrund.

Speziell im Bereich der Expertensysteme lassen sich Erfahrungen, die hier *Fälle* genannt werden, einfach und effizient einsetzen.

Gerade so, wie der menschliche Experte seine konkret gemachten Erfahrungen benutzt, um sie auf die neue Situation zu übertragen, kann ein Expertensystem ebenfalls Lösungsansätze für eine neue Problemstellung aus der Fallbasis inferieren. Die hierbei verwendete Technik bezeichnet man als *fallbasiertes Schließen* (*Case-Based Reasoning* [Kol83], [Sch82]).

Dabei wird in einem ersten Schritt derjenige Fall aus der Fallbasis bestimmt, der der aktuell vorliegenden Situation am ähnlichsten ist. Dessen Lösung wird sodann übertragen und als Ergebnis der vorgelegten Problemstellung betrachtet. Schließlich muß die Güte der gefundenen Lösung überprüft werden.

Besondere Aufmerksamkeit verdient dabei die Tatsache, daß die scharfe Trennung zwischen Wissensakquisitation und -applikation, wie sie in den meisten gängigen Systemen aufscheint, abgeschwächt wird und somit im Vergleich zu menschlichem Lernen angemessener erscheint. Auch die Implementierung des Ansatzes erbringt keine großen Schwierigkeiten, jedoch lassen sich nur gewisse Kategorien von Problemstellungen adäquat repräsentieren. Hierzu gehören in erster Linie die verschiedenen Formen von Diagnosesystemen, insbesondere aus dem medizinischen oder technischen Bereich.

In der Literatur werden häufig andere Begriffe im Zusammenhang mit Case-Based Reasoning gebraucht, so etwa *Memory-Based Reasoning* [SW86] oder auch *Instance-Based Learning* [AKA91]. Eine graphische Interpretation von Ähnlichkeit aus dem Bereich der Musterklassifikation führt zum Begriff der *Nearest Neighbour Classification* [GK79].

Die Lektüre von Arbeiten über fallbasierte Systeme (z.B. [Aha91], [AW92b], [Ric92], [RW91], [Sal88], [Wes93], [WPA92]) wird empfohlen, aber nicht vorausgesetzt.

1.2 Gliederung und Überblick

Ziel der vorliegenden Arbeit ist es, eine Klasse bestimmter *dynamischer Ähnlichkeitsmaße* in Anlehnung an [Wes91] bzw. [Tve77] und deren Lernregeln (*Case-Based Learning*) in fallbasierten Systemen analysierend zu vergleichen und insbesondere die praktische Brauchbarkeit zu ermitteln.

Hierzu werden in einem ersten Teil zunächst die Grundlagen bereitgestellt. Dies geschieht im nachfolgenden Kapitel für fallbasierte Systeme allgemein, wobei besonders auf die im weiteren verwendete Terminologie eingegangen wird.

Anschließend wird im dritten Kapitel auf die Unterklasse fallbasierter Systeme eingegangen, die sich *Case-Based Learning* - Systeme nennen und den eigentlichen Untersuchungsgegenstand dieser Arbeit darstellen. Zum Verständnis des Algorithmus werden die historisch gewachsenen Verbesserungen ausgehend vom einfachen Grundalgorithmus (**CBL1**) bis hin zur elaborierten Version (**CBL4**) im Einzelnen erläutert und bewertet.

Der erste Teil wird durch das vierte Kapitel über *Ähnlichkeit* abgeschlossen, wobei insbesondere psychologische Aspekte eine besondere Berücksichtigung finden, denn die Bewertung der Ähnlichkeit zweier Fälle ist ein zentrales Prinzip fallbasierten Vorgehens. Als das in dieser Arbeit verwendete Ähnlichkeitsmaß wird - gewissermaßen als Synthese der gewonnen Erkenntnisse - das im Patdex/2 System der Universität Kaiserslautern [Wes91] verwendete Schema vorgestellt, welches jedoch an einigen Stellen aufgrund der speziellen Untersuchungs-Anforderungen erweitert worden ist.

Der zweite Teil der Arbeit beschreibt die Testumgebung der dynamischen Lernregeln als *Empirisches Szenario*. Als Grundlage dient dabei eine universelle Fallbasis mit einstellbaren Fallgrößen und Klassendefinitionen, die sich wiederum stark an das Patdex/2 System [Wes91] anlehnt.

Im fünften Kapitel werden die dazu implementierten Module detailliert erläutert und rekapituliert, welche Parameter im Einzelnen untersucht werden und wie die Erfolgskriterien zu spezifizieren sind. Auf besondere Charakteristika der Fallbasen, insbesondere bei den Klassendefinitionen, wird im anschließenden sechsten Kapitel eingegangen. Es zeigt sich hier eine mögliche *Klassifikation* von Fallbasen-Typen, die sich auf die untersuchten Lernstrategien in besonderer Weise auswirken. Ebenfalls werden sowohl das globale Lernverhalten beschrieben als auch lerntechnische Erläuterungen für das Scheitern fallbasierter Adaption geboten.

Der dritte Teil stellt den Kern der vorliegenden Arbeit dar und präsentiert die Untersuchungsergebnisse.

Das siebte Kapitel beschreibt zunächst die Resultate der unterschiedlichen Lernstrategien im statistischen Mittel und kommt zu dem Ergebnis, daß keine signifikanten Vorzüge bestimmter Vorgehensweisen zu verzeichnen sind. Vielmehr lassen sich erst die nachfolgend betrachteten, vom Typ der Fallbasis abhängigen Ergebnisse positiv verwerten: Als Synthese wird in Abschnitt 7.4 ein auto-adaptives Lernschema vorgestellt, das mehrere Lernstrategien parallel benutzt und in Abhängigkeit von den jeweiligen Resultaten einen optimalen Lernprozeß auswählt.

Neben der Zusammenfassung des achten und letzten Kapitels werden im Ausblick Erweiterungs- und Verbesserungsmöglichkeiten des geschilderten Ansatzes ergänzt.

Der erste Anhang A liefert in tabellarischer Form die wesentlichsten statistischen Gesamtresultate, deren prinzipielle Aussagekraft bereits im siebten Kapitel beschrieben worden ist.

Anschließend widmet sich der Anhang B der Implementierung der Testumgebung. Neben einer Beschreibung der Datenstrukturen werden die aus den einzelnen Moduln exportieren Funktionen und Prozeduren erläutert und die bedeutendsten hiervon im Quellcode dargestellt und kommentiert.

Es folgt das Literaturverzeichnis im Anhang C.

Kapitel 2

Grundlagen fallbasierter Systeme

Zunächst werden die wichtigsten Begriffe und Konzepte erläutert, die im Zusammenhang mit fallbasierten Systemen auftauchen und die für die vorliegenden Analysen einzuschränken oder zu abstrahieren waren. Die bedeutendsten Ideen fallbasierter Systeme und Abgrenzungen gegen vergleichbare Bereiche sind anschließend zu betrachten.

2.1 Terminologie

Die Motivation für viele der nachfolgend erläuterten Begriffe leitet sich aus dem Bereich der *Diagnostik* ab. Allerdings finden sie inzwischen auch dort Anwendung, wo konkrete medizinische oder technische Problemstellungen durch andere Aspekte ersetzt worden sind. In Anlehnung an die Fallbasen - Repräsentationssprache *CCRL*[1] des *Inreca*[2]*-Projekts* der Universität Kaiserslautern werden deshalb die allgemeinen, etwas abstrakteren Definitionen angegeben.

2.1.1 Probleme und Merkmale

Ausgangspunkt der Untersuchungen ist ein Universum $\mathcal{U}$ von $\mathcal{N}$-Tupeln als Objekten mit folgender Nomenklatur:

Definition 2.1 (Problembasis, Problem, Merkmal)
Sei $\mathcal{B}$ die Menge aller $\mathcal{N}$-Tupeln mit Werten aus $\mathcal{W} = \{1, 2, ..., \mathcal{M}_{max}\} \cup \{\perp\}$ in jeder Komponente.
$\mathcal{B}$ heißt dann die **Problembasis** *und die Elemente $\mathcal{P} \in \mathcal{B}$* **Probleme der Problembasis** *$\mathcal{B}$ oder kurz* **Probleme**.
Die i-te Komponente von $\mathcal{P}$ bezeichnet man als das i-te **Merkmal** *oder auch das i-te* **Attribut**.

Zur Veranschaulichung dieser Definition betrachten wir den Spezialfall die *Diagnostik*. Hier stellen die Probleme Beschreibungen pathologischer Systeme dar, deren Merkmale als *Symptome* zu bezeichnen sind. Wo es angemessen erscheint, werden beide Begriffe synonym verwendet.

[1] **C**ommon **C**ase **R**epresenting **L**anguage
[2] **In**duktion and **Rea**soning from **Cas**es

Die möglichen Wertebereiche eines Merkmals in der Praxis können sehr verschiedenartig sein. Im human-medizinischen Bereich stellt die *Körpertemperatur* (in Grad Celsius) ein Symptom dar, dessen Ausprägungen prinzipiell reellwertig sind[3], während dagegen das Symptom *Blutgruppe* nur wenige diskrete Werte annehmen kann. Ebenso sind boole'sche Wertebereiche denkbar.

Allerdings beschränken wir uns in der Definition (2.1) auf eine endliche, diskrete Wertemenge für alle Merkmale, mit der bereits sehr viele praktische Situationen modelliert werden können.

Für die Zielsetzung der nachfolgenden Analysen genügt obige Formulierung.

Dem *Wissen* um eine bestimmte Merkmalsausprägung eines Problems kommt eine entscheidende Bedeutung zu. Der Vorgang, jene Information zu ermitteln, bezeichnet man als *Test*, wobei die *Kosten* für diese Untersuchung zu berücksichtigen sind. Solange ein Attributwert noch nicht ermittelt worden ist, ordnet man ihm den Wert *unbekannt* (*unknown*) und das Symbol $\perp$ zu, was die Definition (2.1) rechtfertigt.

2.1.2 Fälle und Lösungen

Im Abschnitt 2.1.1 wurden Probleme als $\mathcal{N}$-Tupel bestimmter bekannter oder unbekannter Merkmale beschrieben. Die Aufgabe eines fallbasierten Systems besteht nun darin, jeder Problemstellung $\mathcal{P}$ eine Menge von Lösungen aus $\mathcal{L}$ zuzuordnen:
$\mathcal{L} := \{L_1, L_2, .., L_n\} \qquad \mathcal{L}_P \subseteq \mathcal{L}.$

Dieses Vorgehen läßt sich als einen *Klassifikationsprozeß* betrachten, der jedem Element der Problembasis mehrere Klassen aus $\mathcal{L}$ zuordnet. (Weitere Angaben hierzu in [Aha91]).

Im allgemeinen lassen sich jedoch nicht für alle Probleme Lösungen angeben. Beispielsweise macht es wenig Sinn, der Aufgabenstellung, die lediglich aus unbekannten Merkmalsausprägungen besteht, eine Lösung zuzuordnen. Dagegen können für eine Problemsituation sehr viele unterschiedliche Lösungen angebracht erscheinen.

Dies wird im Bereich der *Diagnostik*, wo Lösungen *Diagnosen* sind, besonders deutlich: Ein menschlicher Patient kann sowohl eine Appendix-Entzündung als auch eine Grippe haben; ein Auto kann neben einem Motorschaden auch eine defekte Batterie und einen Achsbruch erlitten haben[4].

Als einen *Fall* bezeichnet man ein Problem, dessen Lösungen bereits bekannt sind. Jeder Fall beinhaltet deswegen einen binären *Lösungsvektor*:

Definition 2.2 (Fall, Lösungsvektor, Fallbasis)
Ein **Fall** *ist ein Tupel* $C_k = (\mathcal{P}_k, \vec{l})$*, wobei gilt:* $\mathcal{P}_k \in \mathcal{P}$ *und* $\vec{l}$ *ist der zu* $\mathcal{P}$ *gehörige* **Lösungsvektor**. *Die i-te Komponente dieses Vektors beinhaltet genau dann den boole'schen Wert 1, wenn die Lösung* $\mathcal{L}_i$ *auf* $\mathcal{P}_k$ *zutrifft, ansonsten 0.*
Eine **Fallbasis** *ist eine Menge* $\mathcal{C} = \{C_1, C_2, .., C_m\}$ *von Fällen.*
Die Menge aller Fälle bezüglich einer Problembasis $\mathcal{B}$ *heißt* **vollständig gefüllte Fallbasis** $\mathcal{C}_\mathcal{B}$.

[3] Wir wollen jedoch im Sinne des Patienten hoffen, daß die Temperatur im Intervall [36, 40] zu finden ist!

[4] Spätestens auf der Reparaturrechnung wird dies bestätigt ...

2.1.3 Klassen

In der Praxis sind die Lösungen eines beliebigen Problems im allgemeinen nicht bekannt und es ist gerade Gegenstand der Forschung, wie sich allein aus bereits bekannten Fällen *Konzeptbeschreibungen* finden lassen, mit deren Hilfe ähnliche Probleme lösbar sind [AKA91].

Es wird dabei im Bereich des Case-Based Reasoning zwischen *fallanpassenden* (*case adaption*) und fallvergleichenden (*case matching*) Systemen unterschieden. Obgleich die erstgenannten intuitiv eher als zur Problemlösung geeignet erscheinen, werden wir uns mit reinen fallvergleichenden Algorithmen beschäftigen, weil sich hier spezielle Anforderungen an die Lernregeln ergeben, die der eigentliche Untersuchungsgegenstand dieser Arbeit sind.

Für die Analysen zum Verhalten eines fallbasierten Systems ist es jedoch erforderlich, a priori festzulegen, welche Lösungen auf eine Problemstellung zutreffen, damit überprüft werden kann, ob die im folgenden Kapitel über CBL-Systeme angegebenen Verfahren ein erwünschtes Ergebnis liefern oder nicht. Im Gegensatz zu den ermittelten Lösungen eines Problems werden wir die a priori festgelegten Zuordnungen als *Klassen* bezeichnen. Wenn die Lösungen eines Problems gerade den Klassen entsprechen, war der Lernvorgang *erfolgreich*.

Eine sehr einfache Möglichkeit wäre es, einer bestimmten Menge von Problemen einen zufällig ermittelten Klassenvektor zuzuordnen. Dies ergäbe aber eine sehr unrealistische Fallbasis. Warum etwa sollte ein Fahrzeug mit unter 40 kW und defektem Verteiler dieselbe Diagnose erfahren wie ein LKW mit Auspuffschaden[5]?

Es zeigt sich, daß in der Praxis nur bestimmte *Kategorien* von Fallbasen vorkommen, die keinesfalls als *zufällig entstanden* betrachtet werden können. Vielmehr überträgt sich die inhärente *Struktur* der modellierten Systeme auf die Form der Fallbasis.

Wenn jedoch beliebige Fallbasenstruktuen zugelassen wären, so würden die statistischen Untersuchungsergebnisse verfälscht, weil nur eine kleine Teilmenge der analysierten Fallbasen in der Praxis relevant wären.

Da für fallvergleichende Systeme die Lösungen des ähnlichsten Falles für ein vorgelegtes Problem unverändert übernommen werden, sollte bei der Definition der Klassen stets nur *ähnlichen* Problemen dieselbe Lösungsmenge zugeordnet werden.

Dies geschieht sehr einfach durch die *Interpretation eines Problems als Klassendefinition*. Wenn $\mathcal{P}$ ein beliebiges Problem[6] ist, welches als Definition der Klasse $\mathcal{K}_\mathcal{P}$ dienen soll, so ordnen wir allen anderen Problemen, die lediglich weniger unbekannte Attribute aufweisen, ansonsten jedoch mit $\mathcal{P}$ übereinstimmen[7], ebenfalls die Lösung $\mathcal{K}_\mathcal{P}$ zu. Die Klassendefinition geschieht somit durch *Gruppierung* ähnlicher Probleme um ein spezielles, vorgegebenes $\mathcal{P}$.

Da die Elemente der Problembasis mehreren Klassen angehören können, ergeben sich entsprechend mehrere geforderte Lösungen für ein gegebenes Problem. In der Fallbasis werden allerdings nur solche Fälle aufgenommen, deren Lösungsvektor nicht verschwindet, dessen Komponenten also nicht alle den Wert 0 besitzen.

---

[5] Natürlich ist dies nur dann verwunderlich, wenn nicht beide Fahrzeuge z.B. zusätzlich noch einen Getriebeschaden aufweisen!

[6] ... dessen Merkmale jedoch nicht alle unbekannt ($\perp$) sind !

[7] In 4.3.2 werden wir mit *Übereinstimmung* nicht mehr unbedingt Identität, sondern nur noch starke Ähnlichkeit fordern!

Zur Erläuterung der gerade beschriebenen Klassendefinition betrachte man folgendes Beispiel mit 4 Merkmalen:

Sei $\mathcal{B}$ gegeben als Problembasis mit $\mathcal{M}_{max} := 2$.

Das Problem $\mathcal{P} := \begin{pmatrix} \perp \\ 2 \\ \perp \\ 1 \end{pmatrix}$ sei als Klassendefinition zu $\mathcal{K}_\mathcal{P}$ interpretiert.

Folgende weitere Probleme gehören dann[8] ebenfalls der Klasse $\mathcal{K}_\mathcal{P}$ an und sollten deswegen idealerweise eine gemeinsame Lösung ($\mathcal{L}_\mathcal{P}$) besitzen:

$$\begin{pmatrix} 1 \\ 2 \\ \perp \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ \perp \\ 1 \end{pmatrix} \begin{pmatrix} \perp \\ 2 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} \perp \\ 2 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

Die Auflistung aller Probleme derselben Klasse macht deutlich, daß die Merkmalsausprägung $\perp$ hier nicht als *unbekannt* gedeutet werden kann, sondern vielmehr als *beliebig* (*don't care*). Für die Klassenzugehörigkeit zu $\mathcal{K}_\mathcal{P}$ spielt dieses Attribut demnach keine Rolle und kann jeden Wert annehmen.

Es sei noch einmal daran erinnert, daß die Klassenzugehörigkeit als die *a priori* Lösung eines Problems betrachtet werden kann, deren Übereinstimmung mit der *a posteriori* gefundenen Menge ein Indikator für die Güte des Problemlöseprozesses darstellt. Im praktischen Einsatz von fallvergleichenden Systemen dagegen kann nur der menschliche Benutzer entscheiden, ob eine gefundene Lösung tatsächlich korrekt ist. Diese Rückmeldung wird durch die Klassendefinition simuliert und die Lernreglen werden den Ergebnissen entsprechend optimiert.

2.2 Angrenzende Bereiche

Case-Based Systeme lösen vorgelegte Probleme durch Übernahme des ähnlichsten Falles aus der Fallbasis. Hierbei kann auch ein Adaptionsschritt erfolgen, der die gefundene Lösung entsprechend den Unterschieden zum aktuellen Problem transformiert.

Doch die Betrachtung von fallbasierten Systemen als *problemlösende* Algorithmen ist keineswegs zwingend. Alle Elemente der Fallbasis lassen sich gemäß ihren Lösungen *klassifizieren*. Ein vorgeletes Beispiel ist dann entsprechend den Gegebenheiten in der Fallbasis einzuordnen. Aus dieser Sicht erscheinen Case-Based Systeme als Klassifikatoren, die im Bereich der *Musterklassifikation* eine besondere Anwendung gefunden haben.

Diese Übereinstimmung sollte aber nicht verwundern, weil *jeder* Klassifikationprozeß im weiteren Sinne auch als Problemlöseprozeß erscheint; die Aufgabe besteht ja gerade darin, ein konkretes Muster einzuordnen, gewissermaßen zu lösen.

Umgekehrt muß jede Problemstellung repräsentiert und beschrieben werden, und eben diese Beschreibung kann - wenn schon nicht unbedingt als Muster - so aber sicher als ein zu klassifizierendes Objekt gedeutet werden. Hier ist es dann aber erforderlich, daß die gesuchte Lösung zumindest in impliziter Form bereits in der Wissensbasis zu finden

[8] ... ohne Berücksichtigung der Attribut-Ähnlichkeit (4.3.2)

ist. Arithmetische Probleme etwa werden meistens *konstruktiv* gelöst und hierin liegt ein qualitativer Unterschied. Eine Interpretation dieses Vorgangs führt dann nicht zur Klassifikation.

Fallbasierte Systeme sind demnach mächtiger als reine Klassifikatoren, jedoch können sie nicht als allgemeine Problemlöser betrachtet werden. Ihren besonderen Wert erhalten sie durch die Effizienz des einfachen Vorgehens und die vielseitige Einsetzbarkeit, insbesondere in den verschiedenen Formen technischer und medizinischer *Diagnosesysteme*.

Außerdem stellt die Art des maschinellen Lernens bei der Übernahme gelöster Probleme in die Fallbasis eine einfache und dennoch in hohem Maße adäquate Form der Wissensakquisitation dar. Die Repräsentation des Systemwissens liegt vor in impliziter Form durch die Struktur der Fallbasis, also die Anordnung und Ähnlichkeitsbewertung der Fälle.

Diese aufgeführten Aspekte stellen die wichtigsten Gründe für die immer stärker werdende Bedeutung fallbasierter Systeme dar. Im Folgenden findet jedoch eine Beschränkung der Analysen auf eine bestimmte Teilklasse fallbasierter Vorgehensweisen statt, nämlich die *Case-Based Learning* Systeme.

Kapitel 3

Case-Based Learning Systeme

Der Begriff des *Case-Based Learning* (**CBL**) wird verwendet, um das Augenmerk im *Case-Based Reasoning* auf eine Unterklasse zu lenken, die den Lernaspekt stärker betont und keine Falladaption durchführt. Im wesentlichen führt die Eingabe einer Menge von Trainingsfällen zur Konzeptbeschreibung[1], die nachfolgend vorgeführte Probleme klassifizieren und damit lösen kann.

Ausgehend von einem sehr einfachen Grundalgorithmus **CBL1** wurden verschiedene Verbesserungen vorgeschlagen [Aha91], die bis dato zur der bereits ausgereifteren Version **CBL4** führten, an die sich der für diese Arbeit benutzte Algorithmus stark anlehnt, jedoch mit einigen, nicht unwesentlichen Änderungen.

3.1 Der Grundalgorithmus

Die einfachste Möglichkeit, das Erfahrungswissen durch ein fallbasiertes System zu realisieren, besteht in folgendem Vorgehen (*CBL1*):

1. Speichern der Trainingsfälle als gegebene Fallbasis
2. Ermitteln des zu einem vorgelegten Problem $\mathcal{P}$ ähnlichsten Falles $\mathcal{F}$ in der Fallbasis
3. Übernahme der Klassenzugehörigkeit (Lösung) von $\mathcal{F}$ für $\mathcal{P}$

(vergleiche hierzu auch [Sal88])

Vorgegeben werden also eine Menge von bereits korrekt klassifizierten Fällen, die auch *Trainingsfälle* genannt werden und die somit eine Fallbasis definieren und strukturieren. Jener Fallbasis sind nun Probleme als *Testmenge* vorzuführen.

Auf die Problematik, welche Fälle zum Training geeignet sind, wird noch ausführlicher eingegangen werden müssen. Im allgemeinen besteht die Möglichkeit, jedes klassifizierte Testproblem als neues Datum in die Fallbasis aufzunehmen. Bereits diesen Vorgang darf man als *Lernen* verstehen, weil der neue Fall zusätzliches Erfahrungswissen repräsentiert,

[1] Allerdings liegt diese nicht etwa explizit in symbolischer Form vor, sondern implizit durch die Struktur der Fallbasis

das bei Präsentation weiterer Probleme zu schnellerer oder gar geänderter Lösungsmenge führen kann.

Für die vorliegenden Betrachtungen wird jedoch ein anderer Lernaspekt eine entscheidende Rolle spielen, der sich auf die Schlüsselproblematik der *Fall-Ähnlichkeit* bezieht und der im nächsten Kapitel eingehend behandelt wird. Hier wird das Ähnlichkeitsmaß selbst adaptiert, welches Auswirkungen auf das Klassifikationsverhalten der Fallbasis hat; demgemäß kann auch von *Metalernen* gesprochen werden.

Der Einfachheit des oben erläuterten Vorgehens stehen erwartungsgemäß gravierende Probleme gegenüber, die sich folgendermaßen zusammenfassen lassen:

- Die CBL1 - Algorithmenklasse ist speicher- und rechenintensiv, weil *alle* Trainingsfälle in die Fallbasis aufgenommen werden
- Sie ist sehr anfällig für *verrauschte*, also unvollständige oder fehlerhafte Eingabedaten
- Sie kann irrelevante Merkmale nicht unterscheiden
- Die Wahl des Ähnlichkeitsmaßes wirkt sich sehr stark auf die Güte des Algorithmus aus

David W. Aha erwähnt darüber hinaus die Schwierigkeit, nicht-numerische Merkmalsausprägungen zu bearbeiten und kritisiert die Unbrauchbarkeit der Information, die sich allein aus der Betrachtung der Datenstruktur, also der Fallbasis, ergibt (siehe hierzu [Aha91]).

Es wurden einige Anstrengungen unternommen, die erwähnten Probleme auszuräumen oder doch zumindest zu entschärfen. Details sind in [Aha91] und [Sal88] zu erhalten. Wir werden lediglich einige wesentliche Punkte beleuchten, die zum Verständnis des benutzten Algorithmus erheblichen Beitrag leisten.

3.2 Elaborierte Versionen des CBL-Algorithmus

3.2.1 CBL2

Der erste Verfeinerungsschritt reduziert die Größe der Fallbasis enorm, indem nicht mehr *alle* Trainingsfälle aufgenommen werden, sondern nur noch solche, deren Klassifikation durch die Fallbasis nicht ohnehin korrekt verlaufen wäre. Im Grunde vergleicht man die korrekte Klasse mit der gefundenen Lösung des Algorithmus und nur bei Fehlklassifikation wird der zugehörige Trainingsfall gespeichert. In der Literatur findet sich hierfür die Abkürzung **CBL2**-Algorithmus[2].

Geometrisch läßt sich jenes Vorgehen deuten als Speichern der *Randfälle*, wenn wir die Ähnlichkeit zweier Probleme mit einer gemessenen Entfernung identifizieren (vgl. Kapitel 4). Eine Klasse von Fällen stellt so im Idealfall eine konvexe Menge dar, bei der auf

[2] Hierbei ist anzumerken, daß es sich bei dieser Nomenklatur um eine Klasse von möglichen Algorithmen handelt

innere Elemente verzichtet werden kann. Wären dagegen zur Beschreibung der Klassendefinitionen beliebige prädikatenlogische Formeln zugelassen worden, könnte nicht einmal gewährleistet werden, daß eine Klasse ein zusammenhängendes Gebiet beschreibt!

Allerdings kann dennoch auch ein innerer Punkt falsch diagnostiziert werden, weil die Randfälle nicht *dicht* in dieser diskreten Menge liegen; der Reduktionsschritt ist somit nicht logisch korrekt[3].

3.2.2 CBL3

Um *verrauschte* Eingabedaten besser handhaben zu können, sieht die **CBL3**-Version des Grundalgorithmus neben den oben beschriebenen Änderungen eine weitere Reduktion an Fällen vor. Einzelfälle, die statistisch signifikant häufig nachfolgende Situationen fehlklassifizieren, sind mit großer Wahrscheinlichkeit selbst verrauschte und somit falsche Problembeschreibungen. Sie werden aus der Fallbasis gelöscht, was kognitionspsychologisch als *Vergessen* gedeutet werden mag. Offenbar wirken die Einschränkungen von **CBL2** und **CBL3** kontraproduktiv: Gerade Randfälle, die im **CBL2**-Algorithmus verstärkt in die Fallbasis aufgenommen werden, neigen ja dazu, nachfolgende Situationen der eigenen Diagnose zuzuordnen, obgleich diese Probleme möglicherweise einer benachbarten Klasse angehören. Untersuchungen von [Aha91] und [AKA91] belegen jedoch die Brauchbarkeit dieser Idee.

3.2.3 CBL4

Schließlich müssen die *Bedeutungen* bestimmter Merkmale berücksichtigt werden. **CBL4** gewichtet verschiedene Attribute unterschiedlich, je nach Klassenzugehörigkeit [Aha91]. So kann die Bedeutung des Symptoms *Öltemperatur* bei einer Diagnose *Motorschaden* als wesentlich größer angesehen werden als die des *Reifenluftdrucks*. Gerade umgekehrt verhalten sich die Relationen bei *geplatztem Reifen*.

Im allgemeinen geschieht die Zuweisung bestimmter Attribut-Bedeutungen bei verschiedenen Klassen durch eine *Gewichtsfunktion*.

3.3 Synthese

Für die vorliegende Arbeit wurde eine Sammlung aus den oben dargelegten Ideen zur Verbesserung des *Case-Based Learning* - Algorithmus extrahiert, die sich im wesentlichen an den in [Wes91] beschriebenen Strukturen anlehnt.

Es wird sich zeigen, daß die Verfeinerungen des **CBL4** gänzlich durch die Vorgabe eines Schemas zur Berechnung des Ähnlichkeitsmaßes aufgenommen werden können, das somit anpassungsfähig wird. In [Jan91a] finden sich Untersuchungen, denen zufolge jedes starre Ähnlichkeitsmaß erhebliche Restriktionen aufweist, während der Einsatz dynamischer Maße die Mächtigkeit der CBL-Verfahren auf das Niveau der *Induktiven Inferenz* hebt. Die anderen Vorschläge von **CBL2** bis **CBL4** zur Reduktion des Speicherplatzes wurden den Untersuchungen des Ähnlichkeitsmaßes untergeordnet, jedoch nicht vernachlässigt und als variable Parameter geführt.

[3] Allerdings spielt die Korrektheit in CBL-Systemen eine untergeordnete Rolle, weil in der Praxis stets mit fehlerhaften Eingabedaten gerechnet werden muß

Kapitel 4

Ähnlichkeit

An essay is like a fish![1]
Amos Tversky

Ähnlichkeit ist ein sehr komplexer Begriff. Er wird in völlig unterschiedlichen wissenschaftlichen Bereichen benutzt und seine Ausprägungen variieren entsprechend stark.

Als sehr gefährlich erweist sich die Annahme, die Ähnlichkeit zweier Objekte zueinander sei eine konstante Größe. In [Tve77] findet sich ein einfaches Beispiel: Welchem der drei folgenden Staaten ist *Österreich* am ähnlichsten? *Schweden, Polen* oder *Ungarn*? Die meisten Befragten werden hier wohl als Antwort *Schweden* vorziehen aufgrund der wirtschaftlichen und - bis vor kurzer Zeit - politischen Gemeinsamkeiten. Wird dagegen die Auswahl auf *Schweden, Norwegen* und *Ungarn* geändert, so finden wohl die meisten Angesprochenen die Ähnlichkeit mit Ungarn am größten[2], und dies, obgleich beide Staaten, Ungarn und Schweden, in *beiden* Auswahlmengen vorkamen. Offenbar hängt Ähnlichkeit noch von einem weiteren Parameter ab, nämlich der *Bezugsgröße*.

Eine andere Sichtweise geht davon aus, daß Ähnlichkeit *Gleichheit auf höherer Abstraktionsebene* sei; ihr Wert ergibt sich durch die Anzahl atomarer Abstraktionsschritte. So ist die Ähnlichkeit einer *Buche* gegenüber einer *Fichte* größer als jene zu einer *Rose*, weil es nur den Abstraktionsschritt *Baum-Typ* erfordert, um Buchen und Fichten zu egalisieren, während sich die Abstraktion bis hin zur *Pflanze* als umfassender erweist. Jeder *natürliche* Abstraktionsschritt, der die Buche und die Rose egalisiert, beinhaltet in derselben Menge ebenfalls die Fichte! Deswegen ist die Fichte der Buche *ähnlicher* als die Rose. Hier wird die *absolute* Ähnlichkeit zweier Objekte relativiert auf den Vergleich zu einem dritten. Genaueres siehe [Ric89].

Wir werden uns im folgenden zunächst mit mathematischen Grundlagen von Ähnlichkeitsfunktionen beschäftigen, um dann psychologische Argumente zu analysieren. Schließ-

[1] Tversky beschreibt als Auflösung des Gleichnisses: *At first, the statement is puzzling. An essay is not expected to be fishy, slippery, or wet. The puzzle is resolved when we recall that (like a fish) an essay has a head and a body, and it occasionally ends with a flip of the tail.*

[2] Aufgrund geographischer Nähe!

lich wird sich das von uns verwendete Ähnlichkeitsmaß[3] als Synthese der verschiedenen Ansätze ergeben.

4.1 Ähnlichkeitsfunktionen

4.1.1 Der algebraische Ansatz

Wenn die Ähnlichkeit zweier Objekte einer beliebigen Grundmenge $\mathcal{G}$ gemessen werden soll, so wäre als Ergebnis eine reelle Zahl aus dem Intervall [0, 1] erwünscht. Der Wert 0 würde hierbei bedeuten, daß beide Objekte keine Ähnlichkeit besitzen, während 1 für maximale Ähnlichkeit stünde. Somit handelt es sich bei *Ähnlichkeit* um eine Funktion sim, die folgenden beiden Axiomen genügt: (siehe hierzu [Ric92])

1. $sim(x, x) = 1$ (Reflexivität)
2. $sim(x, y) = sim(y, x)$ (Symmetrie)

Zunächst wird gefordert, daß ein Objekt zu sich selbst die maximale Ähnlichkeit besitzt. Weiterhin sollte es gleichgültig sein, welches von zwei Objekten zuerst betrachtet wird: deren Ähnlichkeit hat in beiden Fällen denselben Wert. Natürlich gibt es eine unüberschaubare Menge von Ähnlichkeitsfunktionen dieser Art und nur wenige erscheinen natürlich. Beispielsweise erfüllt folgende triviale Funktion tri die geforderten Axiome:

1. $tri(x, x) := 1 \quad \forall x \in \mathcal{G}$
2. $tri(x, y) := 0 \quad$ für $x \neq y \quad$ x, y $\in \mathcal{G}$

Dem gegenüber steht die leichte Handhabung von Ähnlichkeitsfunktionen und deren praktische Einsetzbarkeit in vielen, wenn auch nicht allen mathematischen und naturwissenschaftlichen Bereichen.

4.1.2 Der geometrische Ansatz

Es bietet sich an, die Ähnlichkeit zweier Objekte mit einem geometrischen Abstand zu identifizieren. Hierzu ist es erforderlich, alle Elemente in eine mathematische Struktur einzubetten, den *metrischen Raum*, was nichts anderes als eine Menge X von Objekten ist, für die eine *Metrik* d existiert mit:

1. $d(x, y) = 1 \iff x = y$
2. $d(x, y) = d(y, x) \quad \forall x, y \in X$
3. $d(x, z) \leq d(x, y) + d(y, z) \quad \forall x, y, z \in X$

Das dritte Axiom ist die *Dreiecksungleichung*. Eine Unterklasse der Metriken entspricht genau den Ähnlichkeitsfunktionen aus 4.1.1 (Details siehe [Ric92]). Für den Spezialfall

[3] Genauer: das Schema für Ähnlichkeitsmaße

des Raums R^n hat Steven Salzberg [Sal88] den Einsatz bestimmter Metriken und deren Auswirkungen auf die Klassifikationsfähigkeit des CBL-Algorithmus untersucht. Als Ausgangspunkt diente der Ansatz:

$$\Delta_k(x,y) = \sqrt[k]{\sum_{i=1}^{n} |x_i - y_i|^k} \tag{4.1}$$

Für $k = 2$ ergibt sich die *Euklidische Norm*, für $k = 1$ die sogenannte *Manhattan Norm* und schließlich für $k \to \infty$ die *Maximumsnorm*. Auf natürliche Art und Weise induziert eine Norm im Vektorraum eine äquivalente Metrik. In [Sal88] waren die oben genannten Maße Gegenstand verschiedener Untersuchungen von Fallbasen, die jeweils stets in zwei unterschiedliche, disjunkte Klassen einzuteilen waren. Salzberg kommt zu dem Schluß, daß prinzipiell keine der drei Metriken gegenüber den beiden anderen signifikante Vorteile zeigt, jedoch gilt für alle Analysen, daß stets die größere der beiden Klassen in der Fallbasis eine geringere Fehlerrate aufweist.

Mathematische Ansätze eignen sich hervorragend zur Modellierung von Ähnlichkeit in vielen wissenschaftlichen Bereichen, jedoch gibt es auch kritische Beurteilungen der Grundvoraussetzungen, insbesondere von psychologischer Seite. Es wird sich zeigen, daß gerade für das *Case-Based Learning* erst eine Synthese aus mathematischen und kognitiven Gesichtspunkten ein komplexes Ähnlichkeitsmaß erzeugt, das sich sogar dynamisch verändern läßt.

4.2 Psychologische Ähnlichkeit

Bereits in den siebziger Jahren wurde aus psychologischen Überlegungen Kritik am mathematischen Ansatz geübt. Die Dominanz geometrischer Modelle der Ähnlichkeit in theoretischen Analysen wurde zum Anstoß für viele Psychologen. Nach Amos Tverski [Tve77] können keine der drei Voraussetzungen, Reflexivität, Symmetrie und Transitivität, allgemeine Gültigkeit beanspruchen.

- Die **Reflexivität** der Ähnlichkeit kann nicht mehr akzeptiert werden, wenn *Identifikationswahrscheinlichkeit* als Ähnlichkeitsfunktion angesetzt wird. Ein Objekt kann sehr wohl mit einer geringeren Wahrscheinlichkeit korrekt identifiziert als mit einem anderen verwechselt werden. Insbesondere ist die Wahrscheinlichkeit der Identifikation nicht genau 1!

- Die **Symmetrie** der Ähnlichkeit ist nicht zu halten: Ein Sohn mag seinem Vater ähnlich sein, aber kaum wird der Vater mit dem Sohn verglichen. Ebenso mag die Liebe tief wie der Ozean sein, aber kaum wird man die Tiefe des Ozeans mit dem menschlichen Gefühl zu vergleichen suchen. Dies ist kein Zufall. Vielmehr ist der psychologisch wahrnehmbare Ähnlichkeitsbegriff des Menschen faktisch gerichtet und nicht symmetrisch.

- Auch die **Transitivität** gilt im allgemeinen nicht. Mögen Israel und der Iran auch aufgrund der geometrischen Lage noch so große Gemeinsamkeiten aufweisen und sind

auch große Ähnlichkeiten politischer Natur zwischen den USA und Israel festzustellen, so weisen die Vereinigten Staaten und der Iran sicherlich kaum nennenswerte Ähnlichkeiten auf.

Natürlich wurde hier die Vergleichsebene gewechselt; wenn eine Beschränkung auf geometrische Positionen erfolgt wäre, hätte sicherlich kein Gegenbeispiel konstruiert werden können, aber die menschliche Vorstellung von Ähnlichkeit abstrahiert sehr schnell vom eigentlichen Vergleichs-*Feature*!

Tverski schlägt als Alternative ein reines *Feature Matching* vor. Um die Ähnlichkeit zweier Objekte zu messen, setzt man einfach die Menge der gemeinsamen Merkmale den unterschiedlichen gegenüber. Bei gleichen Unterschieden sollte die Ähnlichkeit von Objekten mit Vergrößerung der Gemeinsamkeiten steigen ebenso wie dies durch die Reduktion der unterschiedlichen Features der Fall sein müßte, wenn sich die Menge der Gemeinsamkeiten nicht verringert. Demnach operiert das Maß für die Ähnlichkeit zweier Objekte x und y auf der Mächtigkeit bestimmter *Feature-Mengen*. Wenn wir F als Funktion betrachten, die die Mächtigkeit ihrer drei Argument-Mengen unterschiedlich gewichtet, so ergeben sich an Anforderungen für das Ähnlichkeitsmaß sim:
(wobei $\mathcal{M}_x, \mathcal{M}_y$ und $\mathcal{M}_z$ die zu x, y und z korrespondierenden Merkmale enthalten)

1. **Matching**: $sim(x,y) = F(\mathcal{M}_x \cap \mathcal{M}_y, \mathcal{M}_x - \mathcal{M}_y, \mathcal{M}_y - \mathcal{M}_x)$
2. **Monotonie**: $sim(x,y) \geq sim(x,z)$, falls
   - $\mathcal{M}_x \cap \mathcal{M}_y \supseteq \mathcal{M}_x \cap \mathcal{M}_z$
   - $\mathcal{M}_x - \mathcal{M}_y \subseteq \mathcal{M}_x - \mathcal{M}_z$
   - $\mathcal{M}_y - \mathcal{M}_x \subseteq \mathcal{M}_z - \mathcal{M}_x$

Der erste Vorteil einer solchen Modellierung besteht in seiner universellen Einsetzbarkeit: Quasi jede Objektbeschreibung, ob numerisch oder symbolisch, kann hiermit erfaßt werden. Außerdem ist die Überlegung sehr einfach und intuitiv. Tversky nennt einige weitere Anforderungen an sim, um die psychologische Adäquatheit gewährleisten zu können, die an dieser Stelle jedoch nicht von Interesse sind. Für Einzelheiten siehe [Tve77].

Als eine mögliche Instanz von F, die darüber hinaus auf den Ähnlichkeitsbereich im Intervall [0,1] normiert ist, kann folgende Formel angesetzt werden:

$$S(x,y) = \frac{f(\mathcal{M}_x \cap \mathcal{M}_y)}{f(\mathcal{M}_x \cap \mathcal{M}_y) + \alpha f(\mathcal{M}_x - \mathcal{M}_y) + \beta(\mathcal{M}_y - \mathcal{M}_x)} \quad \alpha, \beta \geq 0, \alpha + \beta > 0 \qquad (4.2)$$

Dies wird von Tversky als das *ratio model* bezeichnet. Gerichtete Ähnlichkeit läßt sich nun durch starke Differenzen zwischen α und β bewerkstelligen.

Der große Vorteil der Überlegung, sich auf Mengen von Eigenschaften zurückzuziehen, führt auf der anderen Seite zu erheblichen Problemen bei inneren Abhängigkeiten der Attributwerte.

Außerdem hat sich gezeigt, daß die Werte von α und β möglicherweise vom speziellen Feature abhängen und nicht global bestimmt werden können. Darüber hinaus sieht das *ratio model* keine adaptive Komponente vor, die die Gegebenheiten einer konkreten Fallbasis stärker berücksichtigen ließe. Wie bereits im vorherigen Kapitel angedeutet, bedeutet dies aus theoretischer Sicht eine erhebliche und nicht annehmbare Einschränkung [Jan91a].

4.3 Synthese: Ähnlichkeit in Patdex/2

Unter Berücksichtigung psychologischer Einwände (4.2) gegen bestimmte mathematische Ähnlichkeitsfunktionen, ergibt sich als ein Ähnlichkeitsmaß, das zur Analyse von CBL-Systemen einsetzbar ist, ein Produkt aus den gewonnenen Erkenntnissen der letzten Abschnitte. Es lehnt sich sehr stark an das im Patdex/2 - Diagnose–System der Universität Kaiserslautern verwendete Schema an. [Wes91]

Die Grundidee besteht nun in der *dynamischen Gewichtung* der unterschiedlichen Symptomausprägungen *in Abhängigkeit zur jeweiligen Klasse*. Damit werden zwei der wesentlichsten Kritikpunkte am **CBL**-Algorithmus behoben, die bereits in 3.1 Erwähnung fanden:

- Irrelevante und besonders wichtige Merkmale können unterschiedlich gewichtet werden (vergleiche hierzu 3.2.3)
- Die Wahl des Ähnlichkeitsmaßes wirkt sich nicht mehr sehr stark auf die Güte des Algorithmus aus, weil die Gewichte flexibel verändert werden können. Die iterative Berechnung einzelner Attribut-Gewichte kann als ein weiterer, höherstehender *Lernprozeß* gedeutet werden, der selbst wiederum das CBL-Lernen steuert.

  Ein derartiger Prozeß kann deswegen auch als *Meta-Lernen* bezeichnet werden.

4.3.1 Die Relevanzmatrix

Als Gewichtsfunktion zur Differenzierung unterschiedlicher Bedeutungen der Merkmale in jeder einzelnen Klasse wird im Patdex/2 - System eine Matrix $\mathcal{R}$ eingeführt, deren $\mathcal{N}$ Zeilen jeweils einem Symptom und deren Spalten den einzelnen m Diagnosen entsprechen. Sie wird gemäß [Wes91] als *Relevanzmatrix* bezeichnet:

$$\mathcal{R} = \begin{pmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{\mathcal{N}1} & \omega_{\mathcal{N}2} & \cdots & \omega_{\mathcal{N}m} \end{pmatrix}$$

Jeder Eintrag in $\mathcal{R}$ stellt ein Indikator für die Bedeutung eines Symptoms in einer bestimmten Diagnose dar. Die Spalten erfüllen folgende Normierungsbedingung:

$$\sum_{i=1}^{\mathcal{N}} \omega_{ij} = 1 \quad \wedge \quad \omega_{ij} \geq 0 \quad \forall j \in \{1, 2, .., m\} \tag{4.3}$$

Die Relevanzmatrix sowie die Normierungsbedingung (4.3) werden für die vorliegende Arbeit übernommen und über den Diagnosesystem-Anwendungsfall hinaus auf die allgemeinere Situation im *Case-Based Reasoning* übertragen.

Die folgenden Ergänzungen in den Abschnitten 4.3.2 und 4.3.3 stellen ebenfalls analoge Erweiterungen der mathematischen Grundlagen des Patdex/2 - Systems dar. Im Einzelfall werden prinzipielle Unterschiede explizit erwähnt.

4.3.2 Die Attribut-Ähnlichkeit

Um die bereits mehrfach geforderte *Abhängigkeit* von Merkmalsausprägungen adäquat zu modellieren, führen wir eine Funktion ζ ein, die die Ähnlichkeit der Attributwerte angibt. In der Praxis ergibt sich diese Ähnlichkeit auf natürliche Weise: ein Patient mit 39,3 Grad Fieber unterscheidet sich weniger von einem Fall mit 39,1 Grad als von einem anderen mit 37,5 Grad. Die Symptomwerte lassen sich hier *ordnen*. Aufgrund der Häufigkeit des Auftretens beschränken wir uns allerdings auf eine *lineare, totale Ordnung*, bei der sich je zwei Werte vergleichen lassen.

Die Eigenschaft, daß die Attributwerte zueinander *ähnlich* sein mögen, kann für den Test in der Fallbasis ein- oder ausgeschaltet werden. Zunächst erfolgt eine Beschreibung der Funktion ζ für den Fall, daß die Ähnlichkeit von Attributwerten berücksichtigt wird. Sie führt zwei Attributwerte als Parameter und ist nicht symmetrisch. Dies ist jedoch nur dann von Bedeutung, wenn einer der beiden Eingabe-Parameter den Wert $\perp$ annimmt. Im Hinblick auf die spätere Anwendung im Ähnlichkeitsvergleich zwischen einem vorgelegten Problem $\mathcal{P}$ und einem Fall aus der Fallbasis wird die Unsymmetrie in der Definition (4.4) verständlich.

Wenn beide Eingabedaten natürliche Zahlen sind, so wird ζ einen um so größeren Wert aus dem Intervall [0, 1] liefern, je geringer die Differenz der beiden Zahlen ist:

$$\zeta(a^i_{\mathcal{P}}, a^i_{Fall}) := \begin{cases} 0 & : \quad a^i_{\mathcal{P}} = a^i_{Fall} = \perp \\ \text{unbekannt} & : \quad a^i_{\mathcal{P}} = \perp \wedge a^i_{Fall} \neq \perp \\ \text{zusätzlich} & : \quad a^i_{\mathcal{P}} \neq \perp \wedge a^i_{Fall} = \perp \\ 1 - \frac{|a^i_{\mathcal{P}} - a^i_{Fall}|}{\mathcal{M}_{max}+1} & : \quad a^i_{\mathcal{P}} \neq \perp \wedge a^i_{Fall} \neq \perp \end{cases} \tag{4.4}$$

(wobei $a^i_{\mathcal{P}}$ für das i-te Attribut des vorgelegten Problems und a^i_{Fall} für das entsprechende Symptom des Falles aus der Fallbasis steht)

In der Terminologie der Klassendefinition sprechen wir genau dann von einem *erfüllten* Attributwert, falls gilt:

$$\zeta(a^i_{\mathcal{P}}, a^i_{Klasse}) \in [\theta, 1] \tag{4.5}$$

(hierbei steht a^i_{Klasse} für die Symptomausprägung des die Klasse definierenden Problems)

Der Wert von θ wurde für alle Tests auf $\theta = 0.75$ fixiert. Sicherlich stellt dies, genau wie die Definition (4.4), eine gewissermaßen willkürliche Festlegung dar. Jedoch wirken die so erhaltenen Fallbasen natürlich und akzeptabel. Darüber hinaus sollen in dieser Arbeit ja keine spezifischen und für eine spezielle Datenbank optimierten Ergebnisse geliefert werden, sondern Gegenstand der Analysen stellen *prinzipielle* Vorgehensweisen zur Adaption des Ähnlichkeitsmaßes dar, deren Qualität zu bewerten ist. In diesem Sinne sind obige Einschränkungen nicht untersuchungsrelevant.

In 6.3.2 wird eingehend begründet, warum auf eine Schwelle wie θ nicht grundsätzlich verzichtet werden kann.

Analog zu (4.5) gelten zwei Attributwerte unter nachfolgender Bedingung als *widersprüchlich*:

$$\zeta(ai_{\mathcal{P}}, ai_{Klasse}) \in [0, \theta[\tag{4.6}$$

Für die Menge der erfüllten Merkmale zwischen einem Problem $\mathcal{P}$ und einem Fall C_k verwenden wir das Symbol $\mathcal{E}_{\mathcal{P},C_k}$, für die widersprüchlichen Merkamle $\mathcal{W}_{\mathcal{P},C_k}$, entsprechend für die unbekannten $\mathcal{U}_{\mathcal{P},C_k}$ und schließlich für die zusätzlichen $\mathcal{Z}_{\mathcal{P},C_k}$. Wenn im Kontext der Bezug klar ist, schreiben wir auch kurz $\mathcal{E}, \mathcal{W}, \mathcal{U}, \mathcal{Z}$. Man beachte, daß nicht alle diese Mengen leer sein können, weil das Problem, welches nur aus $\perp$ (unbekannten) Werten besteht, per definitionem niemals einer Klassendefinition entspricht und deswegen keinen Fall darstellt.

Die Schreibweise C_k^j anstatt C_k deutet an, daß für die Ähnlichkeitsbetrachtung des Falles C_k die Klassenzugehörigkeit $\mathcal{K}_j$ herangezogen wird, also die Attribut-Gewichte der Klasse j verwendet werden. Auch hier wird nach Möglichkeit die kürzere Schreibweise verwendet.

Die soeben beschriebene *Attribut-Abhängigkeit* stellt allerdings, wie eingangs erwähnt, nur eine Alternative zum klassischen Modell dar und ist in der Testumgebung für CBL-Systeme, die im nächsten Kapitel im Detail erläutert wird, abschaltbar. Dann vereinfacht sich ζ zu:

$$\zeta(ai_{\mathcal{P}}, ai_{Fall}) := \begin{cases} \text{unbekannt} & : \quad ai_{\mathcal{P}} = \perp \wedge ai_{Fall} \neq \perp \\ \text{zusätzlich} & : \quad ai_{\mathcal{P}} \neq \perp \wedge ai_{Fall} = \perp \\ 1 & : \quad ai_{\mathcal{P}} = ai_{Fall} \wedge ai_{\mathcal{P}} \neq \perp \wedge ai_{Fall} \neq \perp \\ 0 & : \quad sonst \end{cases} \tag{4.7}$$

Auf diese Weise können die sich ergebenden Unterschiede zwischen einer Attribut-Abhängigkeit[4] und -Unabhängigkeit auf natürliche Art beobachtet und schließlich bewertet werden.

Ergänzend sei hier ein Beispiel für die sich ergebenden Unterschiede zwischen Attribut-Abhängigkeit und -Unabhängigkeit beschrieben. Im Abschnitt (5.1.4.1) über Spezialfälle der Relevanz-Änderung werden wir erneut darauf zurückkommen.

$$\mathcal{K} := \begin{pmatrix} \perp \\ 2 \\ \perp \\ 3 \end{pmatrix} \qquad \mathcal{C} := \begin{pmatrix} \perp \\ 1 \\ \perp \\ 2 \end{pmatrix} \qquad \mathcal{M}_{max} := 4 \tag{4.8}$$

Hierbei sei $\mathcal{K}$ die Klassendefinition und $\mathcal{C}$ ein Fall. Wenn die Attribut-Abhängigkeit nicht betrachtet wird, so widersprechen sich die Symptome 2 und 4 und der Fall $\mathcal{C}$ gehört ergo nicht der Klasse $\mathcal{K}$ an.

Dagegen ändert sich die Situation unter Berücksichtigung der Attribut-Abhängigkeit. Exemplarisch sei die Ähnlichkeit des zweiten Attributwertes berechnet:

[4] Diese Abhängigkeit kann formal auch als *Hintergrundtheorie* der Symptomwerte betrachtet werden

$$\begin{aligned}\zeta(a^2_{\mathcal{C}}, a^2_{\mathcal{K}}) &= 1 - \frac{|a^2_{\mathcal{C}} - a^2_{\mathcal{K}}|}{\mathcal{M}_{max} + 1} \\ &= 1 - \frac{1}{5} \\ &= 0.8 \in [\theta, 1]\end{aligned}$$

Somit gilt das zweite Symptom (und ebenfalls das vierte) als *erfüllt* und $\mathcal{C}$ gehört der Klasse $\mathcal{K}$ an!

4.3.3 Das Ähnlichkeitsmaß

Unter Einhaltung der Nomenklatur für $\mathcal{E}, \mathcal{W}, \mathcal{U}, \mathcal{Z}$ aus 4.3.2 ergibt sich das folgende in der CBL-Testumgebung verwendete Ähnlichkeitsmaß:

$$sim(\mathcal{P}, C^j_k) = \frac{\alpha \sum\limits_{a^i \in \mathcal{E}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C^j_k})}{\alpha \sum\limits_{a^i \in \mathcal{E}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C^j_k}) + \beta \sum\limits_{a^i \in \mathcal{W}} \omega_{ij}(1 - \zeta(a^i_{\mathcal{P}}, a^i_{C^j_k})) + \gamma \sum\limits_{a^i \in \mathcal{U}} \omega_{ij} + \eta|\mathcal{Z}|} \quad (4.9)$$

(vergleiche (6.2) in [Wes91])

Auf die Verwendung von α kann durch folgende Ersetzungen leicht verzichtet werden:

$$\beta' := \frac{\beta}{\alpha} \qquad \gamma' := \frac{\gamma}{\alpha} \qquad \eta' := \frac{\eta}{\alpha}$$

Hierdurch wird α stets auf 1 normiert. Das bereits angesprochene Patdex/2 System [Wes91] belegt für β den Wert 2 und für γ $\frac{1}{2}$, wodurch die Ähnlichkeitsabschätzung etwas pessimistischer wird. Analysen zur Belegung der Parameter β, γ und η finden sich in 7.2.2.

Zur besseren Lesbarkeit von (4.9) wird auch folgende Verkürzung in der Schreibweise angewandt:

$$\begin{aligned}E &= \sum_{a^i \in \mathcal{E}_{\mathcal{P}, C^j_k}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k}) \\ W &= \sum_{a^i \in \mathcal{W}_{\mathcal{P}, C^j_k}} \omega_{ij}(1 - \zeta(a^i_{\mathcal{P}}, a^i_{C_k})) \\ U &= \sum_{a^i \in \mathcal{U}_{\mathcal{P}, C^j_k}} \omega_{ij} \\ Z &= |\mathcal{Z}_{\mathcal{P}, C^j_k}|\end{aligned}$$

Die Formel (4.9) lautet dann:

$$sim(\mathcal{P}, C_k^j) = \frac{\alpha E}{\alpha E + \beta W + \gamma U + \eta Z} \tag{4.10}$$

Die entscheidende adaptive Komponente von (4.9) besteht in der Änderung der Relevanzmatrix $\mathcal{R}$. Hier können entsprechend den Gegebenheiten der Fallbasis Merkmalsbedeutungen gelernt werden. Prinzipiell lassen sich ebenso auch die Parameter α, β, γ und η variieren, was jedoch zu sehr starken Seiteneffekten führt. Änderungen dieser Variablen wirken sich auf das Ähnlichkeitsmaß im Gesamten *global* und also für *jeden* Ähnlichkeitsvergleich innerhalb der Fallbasis aus. Dies ist jedoch unerwünscht. Möglicherweise kann das aktuelle Maß bereits mehrere Diagnosen sehr gut voneinander trennen und nur in einem eingeschränkten Teilbereich ist eine Anpassung noch sinnvoll. Dazu würde es dann in vielen Fällen genügen, die Gewichtung in der Relevanzmatrix für wenige Klassen und nur geringfügig zu ändern.

Weiterhin kann argumentiert werden, daß in einem ersten Schritt die globalen Parameter α bis η zu lernen sind, während eine zweite Stufe anschließend die Relevanzmatrix adaptiert. Jedenfalls scheint die Wirkung einer Änderung in den globalen Parametern im allgemeinen *alle* Klassen zu betreffen und es ist keineswegs offensichtlich, daß hierdurch überhaupt eine bedeutende Wirkung erzielt werden kann.

Dennoch lassen sich vermöge der globalen Parameter verbesserte Lernerfolge erzielen, wie in 7.2.2 gezeigt werden wird.

Weiterhin hat jede Klasse einen Schwellwert δ_j, der das Mindestmaß dafür angibt, daß ein Problem die Lösung des ähnlichsten Falles der Fallbasis übernimmt. Andernfalls gehört die vorgelegte Kombination von Attributwerten möglicherweise zu einer neuen, in der Basis nicht gelernten Klasse. (Einzelheiten in 5.1.3 sowie über die globale Lernwirkung des Schwellwertes in 6.2)

Es ist also sehr genau zu beachten, welche variablen Parameter sich global oder lokal auswirken, und welche sich in der Trainingsphase oder erst in der Testphase bemerkbar machen. Auch wenn zu Vergleichszwecken unterschiedliche Werte für die globalen Parameter und für die einzelnen Schwellwerte angesetzt werden, so sind dennoch hauptsächlich die Einträge der Relevanzmatrix Gegenstand der Lernalgorithmen, die das CBL-System ausmachen.

4.3.4 Initialisierungen

Zum Abschluß dieses Kapitels werden Möglichkeiten zur Initialisierung sowohl der Relevanzmatrix als auch der Schwellwerte für die einzelnen Klassen geliefert.

4.3.4.1 Initialisierung der Relevanzmatrix

Die Definition der Relevanzmatrix aus 4.3.1 läßt offen, welche Startwerte für den Anpassungsalgorithmus als geeignet erscheinen. Für das Patdex/2-System [Wes91] werden hierzu die Vorkommen der einzelnen Symptomausprägungen in Relation zueinander gesetzt. Die bedeutendste Häufigkeit erhält auch entsprechend die größte Initialisierung. Interessant zu beobachten ist jedoch, daß ein *don't care*-Wert in der Klassendefinition keineswegs eine geringe Bedeutung des entsprechenden Merkmals impliziert!

Einige spezielle Testversuche mit verschwindendem Startwert haben bereits nach wenigen Iterationen zu maximaler Bedeutungszuweisung geführt. Der beobachtete Effekt rührt daher, daß eine Diagnose womöglich alle belegten Symptomausprägungen mit verschiedenen anderen Klassen gemeinsam hat, während der einzige Unterschied im entsprechenden $\perp$-Wert liegt. Folgende Initialisierung wurde im wesentlichen aus [Wes91] übernommen:

$$\omega_{ij} = \frac{h_{ij}}{\sum_{i=1}^{\mathcal{N}} h_{ij}} \tag{4.11}$$

wobei h_{ij} gerade die Anzahl derjenigen Fälle mit der Diagnose D_j beinhaltet, deren Symptom $\mathcal{S}_i$ nicht den Wert $\perp$ besitzt, also unbelegt ist.

Natürlich durfte hierzu die Information aus den einzelnen Klassendefinitionen nicht benutzt werden, weil sie in der Praxis gerade den Lerngegenstand darstellen und hier nur als künstliches Konstrukt eingeführt worden sind, um realistische Diagnosezuweisungen an den Problemen durchführen zu können.

4.3.4.2 Initialisierung der Klassen-Schwellwerte

Die maximale Ähnlichkeit eines vorgelegten Problems $\mathcal{P}$ zu einem Fall C_k in der Fallbasis allein genügt nicht, um die Lösungen von C_k auch für $\mathcal{P}$ zu übernehmen. Erst wenn der Mindestwert von δ_j erreicht ist, kann die gefundene Lösung als ausreichend verifiziert angesehen werden[5]. Andererseits wird in der Trainingsphase dieser Schwellwert übernommen, um eine Ähnlichkeit zweier Fälle C_1 und C_2 bezüglich einer Diagnose D_j als zu groß abzuschätzen, falls C_2 de facto nicht D_j als Diagnose besitzt.

Doch welche Werte sollten für die einzelnen δ_j festgelegt werden? Es gibt hierfür grundsätzlich zwei Möglichkeiten. Entweder werden die δ_j zusammen mit den Klassendefinitionen fixiert und im Laufe der Trainings- und Testphase nicht mehr weiter verändert, oder sie sind selbst wiederum Teil eines Lernprozesses. Die Testumgebung erlaubt beide Alternativen.

In [Wes91] wird der Schwellwert als Indikator für die erforderlichen Testkosten angesehen, um die Diagnose D_j zu bestätigen. Er wird zu Beginn der Trainingsphase festgelegt und nicht wieder verändert. Jede Diagnose, deren Verifikation hohe Kosten verursacht, wird demnach mit einem hohen Schwellwert belegt, damit sie bei mehreren zur Auswahl stehenden Kandidaten abgelehnt und eine andere Hypothese bevorzugt wird, deren Überprüfung sehr viel billiger ist[6].

Möglicherweise kann der Schwellwert nutzbringende Informationen über eine Konzeptbeschreibung liefern, wenn er sich aus den Ähnlichkeitswerten der zueinander gehörigen Fälle errechnet. Beispielsweise könnte der Mittelwert aller Ähnlichkeiten den Schwellwert der zugehörigen Klasse initialisieren. Er wäre ein Indikator für die *Streuung* der einzelnen Fälle mit derselben Diagnose. Möglicherweise ist dieser Wert auch zu hoch angesetzt. Als Faustregel kann gelten, daß der Schwellwert das Minimum der Ähnlichkeiten innerhalb der

[5] Allerdings wird zu Testzwecken die Lösung des ähnlichsten Falles dennoch - unter Vorbehalt - übernommen, falls die Ähnlichkeitsfunktion nicht den Wert 0 ergibt!

[6] Beispielsweise sollte im Zweifelsfalle lieber eine Glassicherung im Fahrzeug als die Zylinder-Kopfdichtung ausgewechselt werden ...

jeweiligen Klasse nicht unterschreiten sollte, weil die in 5.1.4 beschriebene Trainingsphase sonst in der Regel nicht mehr terminiert.

Es hat sich gezeigt, daß die doppelte Verwendung von δ_j sowohl in der Traningsphase als auch in der nachfolgenden Testphase nicht nutzbringend sein kann. Vielmehr wird für die Traningsphase einerseits δ_j als Steuerungsmechanismus für das Lernvolumen benutzt, während für die Testphase andererseits ein hiervon unabhänginger Schwellwert benutzt wird, der gemäß [Wes91] als Kostenindikator gelten kann.

(Details auch über die lerntheoretische Bedeutung des Schwellwertes in 6.2 sowie in 7.2.1)

Teil II

Empirisches Szenario

Kapitel 5

Aufbau der Testumgebung

Zur Analyse der dynamischen Lernregeln in *Case-Based Learning* Systemen ist das Ähnlichkeitsmaß (4.9) einzubetten in einen möglichst universell gestalteten empirischen Untersuchungsaufbau, wodurch die erzielten Ergebnisse allgemeine Gültigkeit erhalten.

Selbstverständlich werden die Erkenntnisse der vorangegangenen Kapiteln hierbei Verwendung finden. Im Einzelnen sind folgende Fragen zu klären:

- Welche Strategien zur Gewichtsänderung der Relevanzmatrix sind in der Lernphase optimal?
- Wie wirken sich Änderungen an den globalen Parametern α bis η auf die Klassifikationsfähigkeit des Systems aus?
- Welche Bedeutung kommt der Belegung der Schwellwerte zu?

Zur Klärung dieser Fragen erfolgt zunächst eine konkrete Beschreibung des verwendeten Lernalgorithmus sowie der notwendigen Eingabemöglichkeiten. Sodann werden die zur Untersuchung anstehenden Ähnlichkeits-Parameter rekapituliert und schließlich wird auf das Problem eingegangen, wie die Güte einer gefundenen Lernstrategie geprüft werden kann und welche unterschiedlichen Testkriterien zur Anwendung gebracht worden sind.

Spezifische Erkenntnisse über die Strukturen von Fallbasen und Klassendefinitionen sowie weitergehende Untersuchungen bezüglich veränderter Ansatzmöglichkeiten werden im folgenden Kapitel 6 behandelt.

5.1 Der algorithmische Aufbau

Der verwendete Rahmen für CBL-Systeme teilt sich in folgende Moduln ein:

1. Klassendefinition
2. Erzeugung der Fallbasis
3. Initialisierung der Klassen-Schwellwerte
4. Durchführung der Trainingsphase
5. Abschließende Testphase

Die Grundidee dieses Aufbaus besteht in der Möglichkeit, flexibel und dynamisch alle einstellbaren Parameter und Formen des Ähnlichkeitsmaßes auf naheliegende Weise zu analysieren.

5.1.1 Klassendefinition

Es gibt prinzipiell zwei Arten der Klassendefinition. Die *manuelle Eingabe* gestattet die Modellierung einer beliebigen Fallbasenkonfiguration, deren Verhalten im CBL-Einsatz möglicherweise von gewissem Interesse ist, während die *zufällige Definition* bei einer sehr großen Anzahl von Klassendefinitionen statistische Aussagen über Charakteristika *aller* modellierbaren Fallbasen erlaubt (Resultate hierzu finden sich in Abschnitt 7.1 der Testauswertung).

Über globale Konstanten wird zuvor festgelegt, welche Dimension die Fallbasis schließlich annehmen soll. Neben der Anzahl an Merkmalen, die ja, wie in Abschnitt 2.1.1 beschrieben, für alle Probleme fest ist, muß der diskrete Wertebereich ($\mathcal{M}_{max}$) angegeben werden, der für alle Attribute gleich ist[1].

Zur Rechtfertigung der angeführten Restriktionen sei einerseits an die recht häufig zu beobachtende Willkür bei der Diskretisierung stetiger Merkmalsausprägungen erinnert, während andererseits die statistischen Ergebnisse durch eine Einschränkung des einheitlichen Wertebereichs nur bedingt berührt werden.

5.1.2 Erzeugung der Fallbasis

Die Größe der Fallbasis hängt unmittelbar ab von der Art und Weise der Testdurchführung.

Entweder werden nur die *Randfälle* generiert und in die Datenbank aufgenommen, oder eine zufällige Auswahl von Fällen gelangt in die Fallbasis, wobei über einen globalen Parameter gesteuert wird, wieviel Prozent der klassifizierbaren Fälle hiervon betroffen sind[2]; schließlich besteht die Möglichkeit, *alle* Fälle in die Fallbasis aufzunehmen, um die Klassifizierungsfähigkeit der Relevanzmatrix unter extremen Bedingungen zu testen. Es wird sich zeigen, daß die Trainingsphase unter diesen Umständen häufig nicht mehr terminiert!

Um die Trainingsfälle zu generieren, werden systematisch alle Kombinationen von Symptomwerten als Problemstellungen untersucht und deren Lösungen gemäß den Klassendefinitionen bestimmt; dabei müssen alle belegten Merkmale mit denen der Klasse übereinstimmen, während die unbelegten ($\perp$) einen beliebigen Wert annehmen dürfen (detaillierte Beschreibung in 2.1.3).

Nur für solche Probleme, denen mindestens eine Lösung zuzuordnen ist, schließt sich ein weiterer Entscheidungsprozeß über die Aufnahme in die Fallbasis an. Hierzu ist entweder eine zufällige Wahl mit der durch die globale Variable festgelegten Wahrscheinlichkeit zu treffen, oder die Situation stellt einen *Randfall* einer ihrer Klassen dar.

Letztere Überprüfung geschieht in zwei Stufen: Jede Klassendefinition kann auch als ein Problem $\mathcal{P}_{class}$ interpretiert werden, und zur Ermittlung eines Randfalls spielt die

[1] Im Falle der *zufälligen Klassendefinition* sind lediglich die maximalen Werte der Fallbasenkonfiguration anzugeben, die innerhalb jenes Bereichs wiederum zufällig ermittelt werden

[2] Diese Möglichkeit kommt dem realen Einsatz von CBL-Systemen am nächsten. Je nach Gesamtzahl an Situationen liegt der Anteil der Trainingsfälle weit unter zehn Prozent

Ähnlichkeit zu $\mathcal{P}_{class}$ eine entscheidende Rolle. Jeder Randfall muß die $\perp$-Attribute aus $\mathcal{P}_{class}$ mit einem *extremalen* Wert belegen. Bei Attribut-Abhängigkeit (4.3.2) handelt es sich dabei um die beiden Werte 1 und $\mathcal{M}_{max}$, während anderenfalls jeder numerische Wert als extremal anzusehen ist[3]. Weiterhin müssen die belegten Werte aus $\mathcal{P}_{class}$ übereinstimmen und dies, obwohl für Attribut-Abhängigkeit hier noch weitere Variationsmöglichkeiten bestünden. Allerdings würden dann zu viele Fälle als Randfälle in die Fallbasis aufgenommen werden, was den Effekt der CBL2-Verbesserungen wieder verringerte.

Wenn die Attribut-Abhängigkeit global ausgeschaltet wird, müssen die in $\mathcal{P}_{class}$ belegten Attributwerte *alle* übereinstimmen, da das vorgelegte Problem ansonsten per definitionem (2.1.3) überhaupt nicht der entsprechenden Lösungsmenge zuzuordnen wäre. Dieser Fall läge also bereits jenseits der Klassenabgrenzungen.

Alle Kombinationen aus den geforderten Symptom-Ausprägungen sind somit als Randfälle zu interpretieren. Darüber hinaus wird $\mathcal{P}_{class}$ selbst auch in die Fallbasis aufgenommen.

5.1.3 Festlegung der Schwellwerte

In 4.3.4.2 wurde bereits die Initialisierung der Schwellwerte erwähnt. In der benutzten Testumgebung ist die Belegung der entsprechenden Variablen vor dem Programmstart vorzunehmen, falls sie nicht aus der Fallbasis heraus berechnet werden soll. Welche Werte aber eignen sich als Schwellwerte?

Es hat sich sehr bald herausgestellt, daß die Schwellwerte nicht das Minimum der Fallähnlichkeiten innerhalb einer Klasse annehmen dürfen, weil sie dann sehr niedrige Werte[4] erreichen können und in der Regel die Terminierung der Trainingsphase verhindern. Deshalb wurde als Alternative zu einer fixen Schwelle auf den *Mittelwert* zurückgegriffen. Die *Standard-Abweichung* stellt dann einen Indikator für die Ausdehnung der Klasse dar. Wie in 7.2.1 belegt werden wird, erweist sich eine von der gegebenen Klassensituation abhängige Schwellwertbelegung als wesentlich effizienter als eine fixe Schwelle.

Als weitere Modelle zur Belegung des Schwellwertes sind insbesondere adaptive Verfahren zur Erhöhung der jeweiligen Variable in der Trainingsphase denkbar; sie könnten somit eine *sichere* Terminierung gewährleisten! Auch hierzu finden sich nähere Ausführungen bei der Auswertung der Analyse-Daten in 7.2.1.

Die Interpretation des Schwellwertes als Kostenfaktor, wie in 4.3.4.2 erläutert, wird nur für die Testphase vorgeschlagen, weil sie im praktischen Einsatz von CBL-Systemen hier einen optimalen Nutzen erbringt, dagegen in der Trainingsphase als ungeeignet erscheint. Weiterhin hat sich gezeigt und wird in der Auswertung belegt, daß auch nichtterminierende Trainingsphasen aufgrund zu niedriger Schwellwerte oftmals sehr brauchbare Relevanz-Matrizen erzeugen!

5.1.4 Die Trainingsphase

In der Trainingsphase sind die initialen Werte der Relevanz-Matrix (4.3.4.1) den Strukturen der Fallbasis anzupassen. Insbesondere ist die korrekte Bedeutung der einzelnen

[3] Es gibt hier keine Rechtfertigung für den Vorzug eines bestimmten numerischen Wertes!

[4] Es wurden Ähnlichkeiten unter 0.009 innerhalb einer Klasse beobachtet, aber theoretisch könnte jeder beliebige Wert angenommen werden!

Merkmale in jeder Klasse herauszufinden.

Der entscheidende Untersuchungsgegenstand des Testaufbaus besteht in verschiedenen Möglichkeiten, diesen *Lernprozeß* im Detail zu realisieren und zu bewerten. Hierzu ist jedoch zunächst das prinzipielle Verfahren zur Adaption der Relevanz-Matrix anzugeben.

Dies geschieht durch die Erhebung der Ähnlichkeit derjenigen Fälle, die keine gemeinsamen Diagnosen besitzen. Der Schwellwert dient dabei als Vergleichsmaßstab. Jeweils einer der beiden zu vergleichenden Fälle kann dann als vorgelegtes Problem interpretiert werden, dessen Ähnlichkeit zu dem anderen Fall so groß ist, daß die Lösung akzeptiert werden würde, was allerdings - gemäß der Klassendefinition - falsch wäre[5].

Um diesem Umstand Rechnung zu tragen, ist es geboten, die erfüllten Merkmals-Gewichtungen E in der Relevanz-Matrix zu verringern, während die widersprüchlichen W und unbekannten U zu vergrößern sind. Die zusätzlichen Werte Z bleiben dagegen fest. Denn durch jede dieser Änderungen verringert sich die nominelle Ähnlichkeit der beiden Fälle! Diese prinzipielle Problematik wird in der KI als das *Kreditzuweisungsproblem* bezeichnet.

Aber um welchen Wert sollen die Gewichte verändert werden? Aufgrund der Normierungsbedingung (4.3) ist die Summe der Änderungen an E gleich der Summe der Additionen an $W + U$. Es ist naheliegend, als Heuristik für die Gewichtsänderung folgendem Grundsatz nachzugehen:

Nach der Relevanz-Änderung sollte die Ähnlichkeit der beiden behandelten Fälle gerade auf das Maß des Schwellwertes δ_j zurückgehen.

Es wurden jedoch zu Vergleichszwecken ebenfalls zwei weitere hieran anschließende Strategien getestet, die im folgenden als *Änderungsstrategien* bezeichnet werden. Insgesamt ergaben sich folgende Möglichkeiten, wenn Δ als das Maß für die Größe, um die beide Fälle *zu ähnlich* sind, angesehen wird:

- Die Relevanz-Änderung ergibt eine neue, immer noch zu hohe Ähnlichkeit von $\delta_j + \frac{\Delta}{2}$ (**Less - Strategie**)
- Die Relevanz-Änderung ergibt eine genügende Ähnlichkeit von δ_j (s. Kästchen) (**Normal - Strategie**)
- Die Relevanz-Änderung ergibt eine neue, sehr geringe Ähnlichkeit von $max\{\frac{\delta_j}{2}, \delta_j - \frac{\Delta}{2}\}$ (**More - Strategie**)

Für alle drei Möglichkeiten lassen sich prinzipielle Vorteile finden. Bei einer geringen Änderung der Relevanz-Matrix entspricht der *Lerneffekt* einer einzigen Problem-Präsentation eher dem psychologisch Wahrscheinlichen. Dagegen führt die Änderung im zweiten Punkt das mathematisch Naheliegende aus. Schließlich könnte die dritte Strategie Iterationsschritte der Trainingsphase einsparen, indem sie nachfolgende weitergehende Änderungen bereits vorweggreift. (Die exakten Analysen finden sich in 7.2.3)

[5] Das heißt jedoch keineswegs, daß die vorgelegte Situation *tatsächlich* falsch klassifiziert werden müßte, weil unter Umständen ein anderer Fall mit korrekter Diagnose eine noch größere Ähnlichkeit aufweisen könnte!

5.1.4.1 Herleitung der allgemeinen Relevanz-Änderung

Wir werden im folgenden die Änderungen in der Relevanz-Matrix gemäß der mittleren Strategie herleiten, weil sich hieraus sehr einfach auch die anderen beiden Fälle berechnen lassen. Es wird sich zeigen, daß allein aus der Änderungsstrategie keine eindeutige neue Wertzuweisung für die Gewichte der Relevanzmatrix zu folgern ist. Vielmehr stellt sich die Frage, wie die Gewichtsänderungen auf die erfüllten, widersprüchlichen und unbekannten Attribute zu *verteilen* sind. An einer geeigneten Stelle (5.7) werden deswegen unterschiedliche *Verteilungsstrategien* einzuführen sein.

Gemäß Definition (4.9) kann die Ähnlichkeit eines vorgelegten Problems[6] zu einem Fall folgendermaßen beschrieben werden:

$$sim(\mathcal{P}, C^j_{Casebase}) = \frac{\alpha E}{\alpha E + \beta W + \gamma U + \eta Z} = \delta_j + \Delta \tag{5.1}$$

Die Schreibweise $\delta_j + \Delta$ deutet bereits an, daß die Ähnlichkeit des Problems $\mathcal{P}$ zum Fall $\mathcal{C}_{Casebase}$ unter Berücksichtigung der Klassenzugehörigkeit zu $\mathcal{K}_j$ um den (als positiv angenommenen) Betrag Δ über dem Schwellwert δ_j liegt. Weiter sei vorausgesetzt, daß dies nicht der a priori festgelegten Klassenzugehörigkeit von $\mathcal{P}$ entspräche.

Gesucht sind nun die Änderungen der Gewichte an den erfüllten Merkmalen ($\Delta_\mathcal{E}$), den widersprüchlichen ($\Delta_\mathcal{W}$) und den unbekannten ($\Delta_\mathcal{U}$), so daß gilt:

$$\frac{\alpha(E - \Delta_\mathcal{E})}{\alpha(E - \Delta_\mathcal{E}) + \beta(W + \Delta_\mathcal{W}) + \gamma(U + \Delta_\mathcal{U}) + \eta Z} = \delta_j \tag{5.2}$$

Zur Verkürzung der Schreibweise setzen wir gelegentlich für $E - \Delta_\mathcal{E}$ die Variable E_{Neu} ein, ebenso für $W + \Delta_\mathcal{W}$ einfach W_{Neu} und analog U_{Neu}. Der Wert zusätzlicher Merkmale Z wird nicht verändert, weil es sich hierbei ja gerade um unbelegte Stellen handelt, deren Gewichte in das Ähnlichkeitsmaß nicht eingehen[7]. Somit kann als Ziel dieses Herleitungsprozesses formuliert werden, für E_{Neu}, W_{Neu} und U_{Neu} kompakte Terme zu erhalten, die eine Änderung der Relevanzmatrix und damit den Lernvorgang gestatten.

Es wird vorausgesetzt, daß die Verteilung der Gewichtsänderung zwischen den widersprüchlichen und den unbekannten Ausprägungen folgender, naheliegender Gleichung entspricht:

$$\frac{\Delta_\mathcal{U}}{U} = \frac{\Delta_\mathcal{W}}{W} \tag{5.3}$$

$$\hookrightarrow \Delta_\mathcal{U} = \frac{U}{W} \cdot \Delta_\mathcal{W}$$

$$\hookrightarrow \Delta_\mathcal{U} = c \cdot \Delta_\mathcal{W} \tag{5.4}$$

für $c := \frac{U}{W}$, falls $W \neq 0$ und $U \neq 0$.

Nach Einsetzung von (5.4) in (5.2) und nachfolgender Auflösung der Gleichung ergibt sich:

[6] ... dessen Rolle hier ein Fall aus der Fallbasis übernimmt!

[7] Es wird in Z lediglich die *Häufigkeit des Auftretens* zusätzlicher Attribute gezählt!

$$\Delta_{\mathcal{W}} = \frac{1}{\beta + \gamma \cdot c} \cdot \left[\frac{1 - \delta_j}{\delta_j} \cdot \alpha \cdot E_{Neu} - (\gamma U + \eta Z + \beta W) \right] \tag{5.5}$$

Besondere Aufmerksamkeit verdient hier der Faktor $\frac{1-\delta_j}{\delta_j}$, der für $\delta_j = \frac{1}{2}$ gerade 1 ergibt, für $\delta_j \to 0$ unendlich wird und für $\delta_j = 1$ verschwindet.

Seien nun die alten Gewichte der Relevanz-Matrix mit ω_{ij}^A bezeichnet und analog die neuen Werte mit ω_{ij}^N. Gemäß der Normierungsbedingung (4.3) gilt:

$$\sum_{i=1}^{\mathcal{N}} \omega_{ij}^A = 1 = \sum_{i=1}^{\mathcal{N}} \omega_{ij}^N \tag{5.6}$$

Sei f eine Funktion mit Werten aus [0, 1], die jedem Symptom ein *Verteilungsgewicht* zuordnet, die also etwa bei mehreren erfüllten Symptomausprägungen die gesamte Relevanz-Änderung von E_{Neu} auf die einzelnen Posten verteilt. Dann muß f den folgenden Bedingungen genügen:

$$\sum_{a \in \mathcal{E}} f(a) = 1 \qquad \sum_{a \in \mathcal{W}} f(a) = 1 \qquad \sum_{a \in \mathcal{U}} f(a) = 1 \tag{5.7}$$

Es berechnen sich entsprechend die neuen Gewichte in Abhängigkeit zum Attribut a zu:
(wenn wir statt $\zeta_i(a_{\mathcal{P}}, a_{Fall})$ zur besseren Lesbarkeit kurz ζ_i schreiben)

$$\omega_{ij}^N := \begin{cases} f(a) \cdot \frac{E_{Neu}}{\zeta_i} & : \quad a \in \mathcal{E} \\ f(a) \cdot \frac{W_{Neu}}{(1-\zeta_i)} & : \quad a \in \mathcal{W} \\ f(a) \cdot U_{Neu} & : \quad a \in \mathcal{U} \end{cases} \tag{5.8}$$

Eine naheliegende Möglichkeit für f, die alle Gewichtsänderungen proportional zu den alten Gewichten durchführt, ist folgende:

$$f(a) := \begin{cases} \frac{\zeta_i \cdot \omega_{ij}^A}{E} & : \quad a \in \mathcal{E} \\ \frac{(1-\zeta_i) \cdot \omega_{ij}^A}{W} & : \quad a \in \mathcal{W} \\ \frac{\omega_{ij}^A}{U} & : \quad a \in \mathcal{U} \end{cases} \tag{5.9}$$

Das allgemeine Verfahren zur Konstruktion einer solchen *Verteilungsfunktion* f wird nachfolgend vorgeführt. Wir beschränken uns allerdings - der Intuition entsprechend - auf eine Teilklasse von Verteilungsfunktionen, die berücksichtigen, welchen Wert das Gewicht ω_{ij}^A eines Merkmals bisher angenommen hatte. Gemäß den unterschiedlichen Möglichkeiten zur Wahl von f werden wir auch, wie eingangs erläutert, von verschiedenen *Verteilungsstrategien* sprechen.

Sei $\mathcal{M}$ eine Menge von Merkmalen, für die gemäß (5.7) gelten muß:

$$\sum_{a \in \mathcal{M}} f(a) = 1 \tag{5.10}$$

Sei f_S eine beliebige Funktion. Neben der Identität ($f_S(x) := x$) im bereits erwähnten proportionalen Ansatz (5.9) werden für die Fallbasen-Untersuchungen darüber hinaus eine

konstante Funktion ($f_S(x) := 1$), des weiteren eine polynomiale ($f_S(x) := x^2$) sowie eine exponentielle Funktion ($f_S(x) := e^x$) angesetzt. Auf naheliegende Weise entsteht nun aus f_S die Verteilungsfunktion f:

$$f(a_i) := \frac{f_S(\omega_{ij}^A)}{\sum\limits_{a_k \in \mathcal{M}} f_S(\omega_{kj}^A)} \tag{5.11}$$

Dann gilt:

$$\begin{aligned} \sum_{a \in \mathcal{M}} f(a) &= \sum_{a_i \in \mathcal{M}} \frac{f_S(\omega_{ij}^A)}{\sum\limits_{\hat{a}_k \in \mathcal{M}} f_S(\omega_{kj}^A)} \\ &= \frac{\sum\limits_{a_i \in \mathcal{M}} f_S(\omega_{ij}^A)}{\sum\limits_{\hat{a}_k \in \mathcal{M}} f_S(\omega_{kj}^A)} \\ &= 1 \end{aligned}$$

womit die Gleichung (5.10) erfüllt ist.

Doch nun zurück zur Ausgangsproblematik. Es gilt wegen (5.6), wenn zusätzlich *Rest* definiert ist als diejenige Menge aller Merkmale, die sowohl im vorgelegten Problem als auch im verglichenen Fall den Wert unbekannt, also $\perp$ besitzen:

$$\sum_{a \in \mathcal{E}} \left(f(a) \cdot \frac{E_{Neu}}{\zeta_i} \right) + \sum_{a \in \mathcal{W}} \left(f(a) \cdot \frac{W_{Neu}}{1 - \zeta_i} \right) + \sum_{a \in \mathcal{U}} (f(a) \cdot U_{Neu}) + \sum_{a \in \mathcal{Z} \cup Rest} \omega_{ij}^A = 1 \tag{5.12}$$

Nach Ausklammerung der festen Größen in den Summen ergibt sich:

$$E_{Neu} \sum_{a \in \mathcal{E}} \frac{f(a)}{\zeta_i} + W_{Neu} \sum_{a \in \mathcal{W}} \frac{f(a)}{(1 - \zeta_i)} + U_{Neu} \sum_{a \in \mathcal{U}} f(a) + \sum_{a \in \mathcal{Z} \cup Rest} \omega_{ij}^A = 1 \tag{5.13}$$

Wegen 5.6 und 5.4 gilt weiter (mit ausgeschriebenem W_{Neu} und U_{Neu}):

$$E_{Neu} \cdot \sum_{a \in \mathcal{E}} \frac{f(a)}{\zeta_i} + (W + \Delta_{\mathcal{W}}) \cdot \sum_{a \in \mathcal{W}} \frac{f(a)}{1 - \zeta_i} + (U + c \cdot \Delta_{\mathcal{W}}) + \sum_{a \in \mathcal{Z} \cup Rest} \omega_{ij}^A = 1 \tag{5.14}$$

Wenn die Terme mit $\Delta_{\mathcal{W}}$ zusammengefaßt werden, läßt sich (5.14) so notieren:

$$\Delta_{\mathcal{W}} \cdot \left(\sum_{a \in \mathcal{W}} \frac{f(a)}{1 - \zeta_i} + c \right) + E_{Neu} \cdot \sum_{a \in \mathcal{E}} \frac{f(a)}{\zeta_i} + W \cdot \sum_{a \in \mathcal{W}} \frac{f(a)}{1 - \zeta_i} + U + \sum_{a \in \mathcal{Z} \cup Rest} \omega_{ij}^A = 1 \tag{5.15}$$

Nun läßt sich $\Delta_{\mathcal{W}}$ aus 5.5 einsetzen und der so entstandene Term nach E_{Neu} auflösen:

$$E_{Neu} = \frac{1 + \frac{1}{\beta+\gamma\cdot c}(\beta W + \gamma U + \eta Z)\left(\sum\limits_{a\in\mathcal{W}} \frac{f(a)}{1-\zeta_i} + c\right) - W \cdot \sum\limits_{a\in\mathcal{W}} \frac{f(a)}{1-\zeta_i} - U - \sum\limits_{a\in\mathcal{Z}\cup Rest} \omega_{ij}^{A}}{\frac{\alpha}{\beta+\gamma\cdot c} \cdot \frac{1-\delta_j}{\delta_j} \cdot \left(\sum\limits_{a\in\mathcal{W}} \frac{f(a)}{1-\zeta_i} + c\right) + \sum\limits_{a\in\mathcal{E}} \frac{f(a)}{\zeta_i}} \tag{5.16}$$

Beispielsweise ergibt sich für f aus (5.9), wenn die abkürzende Schreibweise $\chi_{\mathcal{E}} := \sum\limits_{a\in\mathcal{E}} \omega_{ij}^{A}$ und analog $\chi_{\mathcal{W}} := \sum\limits_{a\in\mathcal{W}} \omega_{ij}^{A}$ benutzt wird:

$$E_{Neu} = \frac{\chi_{\mathcal{E}} + \frac{1}{\beta+\gamma\cdot c}(\beta W + \gamma U + \eta Z)\left(\frac{1}{W}\chi_{\mathcal{W}} + c\right)}{\frac{\alpha}{\beta+\gamma\cdot c} \cdot \frac{1-\delta_j}{\delta_j} \cdot \left(\frac{1}{W}\chi_{\mathcal{W}} + c\right) + \frac{1}{E}\chi_{\mathcal{E}}} \tag{5.17}$$

In (5.17) gibt es bei der Berechnung der Änderung also keine unbekannten Werte mehr. Somit wird durch die Festlegung von f auch bereits E_{Neu} eindeutig fixiert. Damit läßt sich mittels (5.5) (sowie 5.4) auch W_{Neu} und U_{Neu} berechnen, womit das Ziel dieser Umformungen erreicht ist.

Sehr leicht lassen sich aus (5.17) auch die beiden anderen Änderungsstrategien ermitteln, dies geht unmittelbar aus den Strategie-Beschreibungen hervor und bedarf keiner weiteren Erläuterung.

5.1.4.2 Spezialfälle der Relevanz-Änderung

Die obige, allgemeine Herleitung gilt nur, wenn die Mengen $\mathcal{E}, \mathcal{W}$ und $\mathcal{U}$ alle nicht leer sind, was in Gleichung (5.4) vorausgesetzt werden mußte. Folgende Möglichkeiten sind allerdings in Betracht zu ziehen:

1. $\mathcal{W} \neq \emptyset \wedge \mathcal{U} \neq \emptyset$
2. $\mathcal{W} \neq \emptyset \wedge \mathcal{U} = \emptyset$
3. $\mathcal{W} = \emptyset \wedge \mathcal{U} \neq \emptyset$
4. $\mathcal{W} = \emptyset \wedge \mathcal{U} = \emptyset$

$\mathcal{E}$ kann nicht leer sein, weil dies ansonsten eine Ähnlichkeit von 0 implizieren würde, die kein positives δ_j überschreitet[8]!

- Der erste obige Punkt wurde bereits im vorangegangenen Abschnitt behandelt.

[8] ... und δ_j wird stets als positiv angesetzt!

- Im zweiten Fall vereinfacht sich die Gleichung (5.5) zu:

$$\Delta_{\mathcal{W}} = \frac{1}{\beta} \cdot \frac{1-\delta_j}{\delta_j} \cdot \alpha \cdot E_{Neu} - \frac{\eta}{\gamma} Z - W \tag{5.18}$$

Folgerichtig ergibt sich anstatt (5.16):

$$E_{Neu} = \frac{\beta \left(1 - \sum\limits_{a \in \mathcal{Z} \cup Rest} \omega_{ij}^{A}\right) + \eta Z \cdot \sum\limits_{a \in \mathcal{W}} \frac{f(a)}{1-\zeta_i}}{\frac{1-\delta_j}{\delta_j} \cdot \alpha \sum\limits_{a \in \mathcal{W}} \frac{f(a)}{1-\zeta_i} + \beta \sum\limits_{a \in \mathcal{E}} \frac{f(a)}{\zeta_i}} \tag{5.19}$$

Nach Ersetzung von f aus (5.9) berechnet sich $\Delta_{\mathcal{E}}$ zu:

$$\Delta_{\mathcal{E}} = E \cdot \frac{\frac{1-\delta_j}{\delta_j} \cdot \alpha E - \eta Z - \beta W}{\frac{1-\delta_j}{\delta_j} \cdot \alpha E + \frac{\chi_{\mathcal{E}}}{\chi_{\mathcal{W}}} \cdot \beta W} \tag{5.20}$$

Für den Spezialfall, daß sowohl $\alpha = \beta = \eta = 1$, die Schwelle den Wert $\delta_j = \frac{1}{2}$ besitzt und darüber hinaus $\chi_{\mathcal{E}} = \chi_{\mathcal{W}}$ gilt, entsteht eine sehr kompakte Form von (5.20):

$$\Delta_{\mathcal{E}} = E \cdot \frac{E - Z - W}{E + W} \tag{5.21}$$

- Falls, wie im dritten Punkt aufgelistet, keine widersprüchlichen Symptome auftauchen, genügt E_{Neu} folgender Gleichung:

$$E_{Neu} = \frac{\gamma \left(1 - \sum\limits_{a \in \mathcal{Z} \cup Rest} \omega_{ij}^{A}\right) + \eta Z}{\frac{1-\delta_j}{\delta_j} \cdot \alpha + \gamma \sum\limits_{a \in \mathcal{E}} \frac{f(a)}{\zeta_i}} \tag{5.22}$$

Analog zu (5.20) ergibt sich:

$$\Delta_{\mathcal{E}} = E \cdot \frac{\frac{1-\delta_j}{\delta_j} \cdot \alpha E - \eta Z - \gamma U}{\frac{1-\delta_j}{\delta_j} \cdot \alpha E + \chi_{\mathcal{E}} \cdot \gamma} \tag{5.23}$$

- Der vierte Punkt ist problematischer. Es kann tatsächlich passieren, daß zwei Fälle keine gemeinsamen Lösungen besitzen, obgleich nur erfüllte und zusätzliche Merkmale aufscheinen.

  Folgendes Beispiel (mit vier Attributen) sollte dies veranschaulichen:

  Sei als Klasse gegeben: $\mathcal{K} := \begin{pmatrix} \perp \\ 2 \\ \perp \\ 3 \end{pmatrix}$. Der Fall $\mathcal{C} := \begin{pmatrix} \perp \\ 1 \\ \perp \\ 2 \end{pmatrix}$ entspricht also gemäß (2.1.3) den Anforderungen zur Klassenzugehörigkeit, falls die *Attribut-Abhängigkeit*

aus 4.3.2 vorausgesetzt wird (siehe Beispiel (4.8)!). Betrachten wir nun das Problem $\mathcal{P} := \begin{pmatrix} 1 \\ 1 \\ \perp \\ 1 \end{pmatrix}$, das gegenüber $\mathcal{C}$ nur erfüllte oder zusätzliche Merkmale aufweist. $\mathcal{P}$ gehört jedoch per definitionem (2.1.3) nicht der Klasse $\mathcal{K}$ an.

Es war möglich, dieses Beispiel zu konstruieren, ohne auf Schwellwertbetrachtungen oder Gewichte in der Relevanz-Matrix $\mathcal{R}$ Bezug zu nehmen. Demnach steckt die Schwierigkeit inhärent im gesamten Ansatz und kann ohne weiteres nicht behoben werden.

Wenn im Laufe der Trainingsphase eine derartige Konfiguration aufscheint, werden keine Relevanz-Änderungen vorgenommen, sondern lediglich die Vorkommnisse solcher Anforderungen gezählt. Wir sprechen dann auch von einem *Impossible Requirement* (*IR*).

Ein *IR* kann durch anderweitige Änderungen in der Relevanz-Matrix entstehen oder auch verschwinden. Die Terminierung der Trainingsphase geschieht jedoch unabhängig vom Auftreten der *IR*'s. Sie stellen einen Indikator für sehr *dichte* Klassendefinitionen dar, bei der sich Diagnosen stark überschneiden. So gibt es im obigen Beispiel nicht viele Lösungen für $\mathcal{P}$, die nicht auch auf $\mathcal{C}$ zutreffen[9].

Angemerkt sei, daß die *Attribut-Unabhängigkeit* (4.3.2) das Auftreten von *IR*'s stets verhindert. Hier impliziert ein Problem aus nur erfüllten - also gleichen - Attributwerten und zusätzlichen dieselbe Klassenzugehörigkeit!

Den Prozeß, alle Kandidaten aus der Fallbasis gegeneinander zu vergleichen und die Relevanz-Matrix entsprechend zu verändern, bezeichnet man als einen *Iterationsschritt*, oder kurz als *Iteration*. Die maximale Anzahl an Iterationen in der Testumgebung wird über eine globale Konstante festgelegt[10]. Wenn dieser Wert erreicht wird, dann hat die Trainingsphase nicht *terminiert*. Die Anzahl an Iterationen bis zur Terminierung kann andernfalls als ein Kriterium zur Bewertung der Güte eines Ähnlichkeitsmaßes angesehen werden.

5.1.5 Die Testphase

Nach der Durchführung der Trainingsphase wird die Relevanz-Matrix, und damit das gelernte Ähnlichkeitsmaß getestet. Wir unterscheiden folgende Kategorien von Tests:

- Testsituationen mit Informationsverlust
  - mit vollständig gefüllter Fallbasis
  - mit partieller Fallbasis (Zufallsauswahl)
  - ausschließlich mit Randfällen (*bco*[11])

[9] Allerdings gibt es sie dennoch! Ansonsten wäre es kein Gegenbeispiel ...
[10] Der Wert 20 hat sich in den meisten untersuchten Fällen als ausreichend erwiesen!
[11] **b**oundary **c**ases **o**nly

- Testsituationen ohne Informationsverlust
  - mit vollständig gefüllter Fallbasis
  - mit partieller Fallbasis (Zufallsauswahl)
  - mit bco

Über eine globale Variable des Testszenarios kann zwischen *Zufallsauswahl* der Fälle und *Auswahl der Randfälle* getrennt werden. Im ersten Fall ist zusätzlich einzustellen, mit welcher *Wahrscheinlichkeit* ein (gelöstes) Problem in die Fallbasis aufgenommen wird. Selbstverständlich entspricht dabei einer Wahrscheinlichkeit von 1.0 eine vollständig gefüllte Fallbasis, die somit nur einen Spezialfall partieller Füllung darstellt.

Eine weitere globale Variable steuert den prozentualen Informationsverlust in den einzelnen Problembeschreibungen. Eine Verlustwahrscheinlichkeit von 0.0 entspricht gerade den vorgeführten Fällen ohne Informationsverlust. Alle möglichen Problembeschreibungen mit nicht-leerem Diagnosevektor werden während des Tests der Reihe nach generiert und jeder Attributwert wird mit der geforderten Wahrscheinlichkeit auf $\perp$ gesetzt. Anschließend wird das so entstandene Problem $\mathcal{P}$ der Fallbasis zur Lösung vorgelegt, wobei die Ähnlichkeit von $\mathcal{P}$ zu den einzelnen Fällen $\mathcal{C}_k$ errechnet wird. Für jede Klasse von $\mathcal{C}_k$ ist dabei eine eigene Ähnlichkeitsbewertung entsprechend den Merkmalsbedeutungen der Relevanzmatrix vorzunehmen. Eine korrekte Klassifikation liegt vor, wenn die gefundene Lösung des ähnlichsten Falles[12] im Diagnosevektor des vorgelegten Testproblems auftaucht.

Die Klassifikationskorrektheit kann nach der Testdurchführung mit den Wahrscheinlichkeiten für verrauschte Eingaben[13] und dem Anteil an Trainingsfällen verglichen werden. Die Ergebnisse stellen somit ein wichtiges Kriterium zur Bewertung der Güte des verwendeten Ähnlichkeitsmaßes dar. (Eine graphische Darstellung dieser Resultate findet sich in 7.1)

Angemerkt sei, daß die Austestung einer vollständig gefüllten Fallbasis ohne Informationsverlust sehr wohl sinnvoll ist. Natürlich muß als Ergebnis eine Korrektheit von 100 % zu erwarten sein, wenn die Trainingsphase terminiert; im anderen Fall wird sich jedoch zeigen, wie gut eine Relevanz-Matrix konditioniert ist, wenn die Trainingsphase nicht erfolgreich verlaufen ist.

5.2 Spezifikation der untersuchten Parameter

Aufgrund der Vielzahl an verwendeten Variablen und den sich dadurch ergeben Parametern wird an dieser Stelle zusammengefaßt, welche Möglichkeiten zur Analyse der Ähnlichkeitsmaße offen stehen und welche Einstellmöglichkeiten andererseits nicht verändert werden.

Das Ähnlichkeitsmaß (4.9) verfügt über die vier global wirkende Parameter α, β, γ und η, wovon auf α, wie gesehen, prinzipiell verzichtet werden könnte. Selbstverständlich lassen sich nicht alle Kombinationen dieser Einstellungen testen.

[12] Die Ähnlichkeit muß allerdings den Wert 0.0 überschreiten!

[13] genauer: Eingaben mit Informationsverlust

Neben der gleichen Belegung aller vier globalen Parameter wurden jeweils Unter- sowie Überbewertungen aller Variablen vorgenommen. Folgende Liste verdeutlicht das Untersuchungsvolumen:

Nr.	α	β	γ	η	Bez.
1.	1.0	1.0	1.0	1.0	eq
2.	1.0	10.0	1.0	1.0	$+\beta$
3.	1.0	0.1	1.0	1.0	$-\beta$
4.	1.0	1.0	10.0	1.0	$+\gamma$
5.	1.0	1.0	0.1	1.0	$-\gamma$
6.	1.0	1.0	1.0	10.0	$+\eta$
7.	1.0	1.0	1.0	0.1	$-\eta$
8.	1.0	10.0	10.0	10.0	$-\alpha$
9.	1.0	0.1	0.1	0.1	$+\alpha$

$+\alpha$ und $-\alpha$ wird durch eine gleichmäßige Veränderung der jeweilig anderen Parameter simuliert. Das Auswertungsergebnis findet sich in 7.2.2. Da die Variablen *globale* Wirkung zeigen, wird von einem adaptiven Ansatz zur Einstellung vorerst abgesehen.

Die Initialisierung der Relevanzmatrix wurde aus [Wes91] übernommen und im wesentlichen beibehalten. Es sei jedoch hinzugefügt, daß die Relevanzmatrix auch dann in den jeweiligen Klassen für jedes Merkmal einen Wert erhält, wenn, aufgrund von sehr geringen Trainingsanteilen, kein Fall in der entsprechenden Lösungsklasse zu finden ist.

Als mögliche Änderungsstrategien werden gemäß 5.1.4 untersucht:

1. Änderung der Relevanz, so daß die Ähnlichkeit der aktuellen Fälle gerade auf δ_j zurückgeht (**Normal**)
2. Eine deutlich geringere Änderung (**Less**)
3. Eine deutlich umfangreichere Änderung (**More**)

Außerdem werden folgende Verteilungsstrategien analysiert, die für die Verteilung der Relevanzen auf die neuen Gewichte verantwortlich sind:

1. $f_S(x) := 1$ (**konstant**)
2. $f_S(x) := x$ (**proportional**)
3. $f_S(x) := x^2$ (**polynomial**)
4. $f_S(x) := e^x$ (**exponentiell**)

5.3 Erfolgskriterien

In 5.2 wurden die verschiedenen Parametereinstellungen aufgeführt, die Gegenstand der Analysen waren. An dieser Stelle wird gezeigt, welche *Kriterien zur Beurteilung* der verschiedenen Lernstrategien und Einstellungen dienen. Es ist jedoch nicht in jedem Falle offensichtlich, was ein optimales Resultat darstellt, weswegen die Beurteilungskriterien hier lediglich als Fragen formuliert bleiben. Ausführliche Begründungen finden sich im Kapitel 7 der Testauswertung.

Sicherlich stellt der *Trainingserfolg* ein wesentlicher Bestandteil zur Beurteilung einer Lernstrategie dar. Hat die Trainingsphase mit stabilen Gewichten terminiert?

Hieran schließt sich sofort die Frage an: Wieviele Iterationen wurden benötigt und wie groß ist die Zahl an Änderungen in der Relevanz-Matrix? Interessanterweise wird sich zeigen, daß weder eine sehr große, noch eine sehr geringe Zahl an Änderungen aus lerntheoretischer Sicht als besonders geeignet erscheint. Hierauf wird in 7.1 eigens eingegangen werden.

Welche Schlußfolgerungen können aus den Ähnlichkeits-Mittelwerten und den zugehörigen Standardabweichungen für die Belegung der globalen Konstanten und der Klassenschwellwerte gezogen werden?

Oftmals läßt sich allein anhand der resultierenden Relevanzmatrix unmittelbar nach der Trainingsphase ablesen, ob der Lernvorgang erfolgreich gewesen ist. Dies ist auf die Möglichkeit zurückzuführen, die Klassendefinitionen a posteriori mit den Merkmals-Relevanzen zu vergleichen, was im praktischen CBL-Einsatz natürlich nicht möglich ist[14]!

Wir werden sehen, daß nicht immer die korrekte Bedeutung der Merkmale aus der Klassendefinition abzulesen ist. Deswegen wird die Güte der Matrix auf die Probe gestellt: Testfälle werden mit Informationsverlust behaftet (5.1.5). Wie gut klassifiziert eine Relevanzmatrix in den verschiedenen Informations-Verlust-Stufen (0 % bis 100 % in 10 %-Schritten)?

Darüber hinaus ist es interessant zu beobachten, in welchem Umfang die Klassifikationsfähigkeit bei geringeren Traingsanteilen (10 % bis 100 % in 10 %-Schritten, sowie 5 % und bco) sinkt, und zwar in Abhängigkeit der verschiedenen Parameter!

Insgesamt sind also an möglichen Testkombinationen für jede generierte Fallbasis durchführbar:

[14] Sonst könnte man sich den Lernprozeß vereinfachen, indem man sofort aus der Klassendefinition die richtigen Relevanzen *abgelesen* würde.

- 2 Schwellwerteinstellungen (manuell[a] oder automatisch[b])
- 9 Einstellungen für die globalen Parameter
- 4 mögliche Verteilungsstrategien
- 3 Änderungsstrategien
- 12 Stufen von Traingsanteilen
- 11 Einstellungen mit Informationsverlust

[a] Repräsentiert durch einen festen Wert von 0.7 (zu Vergleichszwecken)
[b] Als Mittelwert der Fallähnlichkeiten innerhalb einer Klasse. Eine weitere Verfeinerung ist in 7.2.1 dokumentiert

Es ergeben sich somit an Untersuchungsvolumina $2 \cdot 9 \cdot 4 \cdot 3 \cdot 12 \cdot 11 = 28512$ Tests pro Fallbasis. Aufgrund dieses Umfangs wurden die Prioritäten zur Analyse der Einzelparameter auf folgende Schwerpunkte gelegt:

- Jeder eingestellte Test beinhaltet *alle* Trainingsstufen und *viele* Grade an Informationsverlust
- Die Änderungs- und Verteilungsstrategien stellen den Hauptuntersuchungsgegenstand dar und werden in allen Kombinationen der anderen Einstellungen stets getestet
- Schwellwert und globale Parameter besitzen eine geringere Priortät

5.4 Zusammenfassung

In diesem Kapitel wurde der Testaufbau beschrieben, der sich im wesentlichen an die Strukturen der CBL-2 bis CBL-4 Systeme und besonders an Patdex/2 [Wes91] orientiert.

Einige Unterschiede ergeben sich durch mehrfache Klassenzugehörigkeiten, die in oben erwähnten Systemen nicht vorgesehen sind. Die Resultate können dennoch als gültig für die meisten *Case-Based Learning* Verfahren angesehen werden.

Den Hauptuntersuchungsgegenstand stellt das Ähnlichkeitsmaß aus (4.9) dar, dessen Strategien zur Relevanz-Änderung zu untersuchen und zu bewerten sind. Aus der Sicht maschinellen Lernens kann dieses Ähnlichkeitsmaß ebenfalls als *dynamische Lernregel* gedeutet werden. Daneben sollen Erkenntnisse über die global wirkenden Parameter sowie die Bedeutung der Schwellwerte gewonnen werden.

Kapitel 6

Charakteristika von Fallbasen

Bei der Analyse von Lernregeln in CBL-Systemen wurden spezielle Erkenntnisse über die Strukturen von Fallbasen und den zugrundeliegenden Klassendefinitionen gewonnen - gewissermaßen als Nebenprodukt - und weitergehende Ansatzmöglichkeiten überprüft. Jenen *Charakteristika* widmet sich das aktuelle Kapitel.

6.1 Klassencharakteristika

Um die Testergebnisse in adäquater Form interpretieren zu können, ist es erforderlich, die verschiedenen Typen von Klassendefinitionen gemäß ihren wesentlichen Eigenschaften zu unterscheiden. Hierzu stehen verschiedene Möglichkeiten bereit, die im folgenden detaillierter beschrieben werden sollen.

6.1.1 Überlastung von Merkmals-Relevanzen

Eine graphische Interpretation der Klassifizierungsaufgabe im fallbasierten System führt zu einem Problem, welches im Gebiet des Konnektionismus [Ric90] hinlänglich bekannt ist und im folgenden als *XOR-Problem* bezeichnet wird.

Abbildung 6.1: Klassenüberschneidungen

Sei wie in Abbildung 6.1.1 eine Fallbasis mit vier Fällen gegeben, wobei stets jene Probleme als eine Klasse betrachtet werden, deren beiden Merkmale entweder ungleich (Klasse 1) oder gleich (Klasse 2) sind. Dann ist es nicht möglich, den einzelnen Attributen eine Relevanz beizumessen, die zur korrekten Trennung der Klassenelemente führt. Die Trainingsphase bei einer solchen Klassendefinition wäre also nicht erfolgreich.

Aus diesem Grunde, um das klassische XOR-Phänomen zu vermeiden, wurde in 2.1.3 eine Klassendefinition gewählt, die per sese diese Problematik vermeidet, weil eine Situation wie in Abbildung 6.1.1 nicht mehr formulierbar ist: Fälle können nur durch eine bestimmte Kombination von Merkmalen klassifiziert werden, und nicht durch das *Verhältnis* bestimmter Merkmalsausprägungen. Im letzteren Fall ist es erforderlich, zusätzliche, neue Klassen einzuführen, die dann auf einer höheren Abstraktionsebene als wiederum zueinander gehörig betrachtet werden mögen. Für die *Trennbarkeit* von Klassen in CBL-Systemen scheint somit eine prinzipielle Beschränkung überwunden.

Es hat sich jedoch gezeigt, daß die XOR-Schwierigkeit auf einer völlig neuen Ebene, unter etwas veränderten Voraussetzungen, wiederum auftritt und hier - wie im Falle des Konnektionismus - prinzipiell nicht lösbar ist.

Man betrachte als einführendes Beispiel die folgenden drei Klassendefinitionen:

$$\mathcal{K}_1 := \begin{pmatrix} 1 \\ 2 \\ \bot \\ \bot \\ \vdots \\ \bot \end{pmatrix} \qquad \mathcal{K}_2 := \begin{pmatrix} 2 \\ 2 \\ \bot \\ \bot \\ \vdots \\ \bot \end{pmatrix} \qquad \mathcal{K}_3 := \begin{pmatrix} 2 \\ 1 \\ \bot \\ \bot \\ \vdots \\ \bot \end{pmatrix} \tag{6.1}$$

Die genaue Anzahl der Attribute und deren Wertebereich ist unerheblich. Was geschieht in der Trainingsphase mit den Fällen aus den drei Klassen, welches Lernverhalten zeigt die Relevanzmatrix?

Offensichtlich unterscheiden sich alle Probleme der Klassen $\mathcal{K}_1$ und $\mathcal{K}_2$ - im statistischen Mittel - nur im ersten Symptom, während das zweite stets übereinstimmt. Für die Änderung in der Relevanzmatrix bedeutet dies - ganz gleich, welche Lernstrategie angewendet wird - daß die Bedeutung des zweiten Merkmals bei allen Veränderungen zurückgeht, und zwar - wiederum im statistischen Mittel, weil auch andere Symptome unterschiedlich sein mögen - stets zugunsten des ersten Merkmals. Gäbe es also im fallbasierten System nur diese beiden Klassen, so würde die Trainingsphase sicherlich bald mit dem Ergebnis terminieren, daß die Bedeutung des ersten Attributs sehr groß wird[1].

Allerdings ernüchtert ein Vergleich der beiden Klassen $\mathcal{K}_2$ und $\mathcal{K}_3$ sehr bald: Der hier auftretende Effekt wirkt gerade gegen den oben beschriebenen. Die Bedeutung des ersten Merkmals verschwindet auf Kosten des zweiten, und zwar - im Mittel - um genau den Wert, der im Vergleich von $\mathcal{K}_1$ und $\mathcal{K}_2$ zu verzeichnen war. Auch die durch die Ähnlichkeitsbetrachtung der Fälle aus den Klassen $\mathcal{K}_1$ und $\mathcal{K}_3$ gewonnene Lerninformation verhilft aus der Pattsituation nicht hinaus: Hier sind die Bedeutungen der beiden ersten Attribute exakt gleich hoch!

Welche Reihenfolge im Vergleich der einzelnen Fälle eingehalten wird, hat zwar eine große Auswirkung auf die letztlich übernommenen Werte in der Relevanzmatrix, allerdings

[1] In Abhängigkeit vom Schwellwert und der Anzahl an Merkmalen zwischen 0.6 und 0.999

kann durch keine noch so ausgereifte Lernstrategie dieses prinzipielle Hindernis überwunden werden; die gegenläufigen Lerneffekte bedingen, daß die Trainingsphase keinesfalls[2] terminiert.

Nur bei sehr tiefgründiger Betrachtung des oben beschriebenen Phänomens werden die Parallelen zur XOR-Problematik deutlich. Selbstverständlich stellt das dargelegte Beispiel lediglich eine Minimalkonstellation für die auftauchende Schwierigkeit dar. Eine leichte Erweiterung führt zu folgender Ausgangsbasis:

$$\mathcal{K}_1 := \begin{pmatrix} 1 \\ 2 \\ 2 \\ \bot \\ \bot \\ \vdots \\ \bot \end{pmatrix} \qquad \mathcal{K}_2 := \begin{pmatrix} 2 \\ 2 \\ 1 \\ \bot \\ \bot \\ \vdots \\ \bot \end{pmatrix} \qquad \mathcal{K}_3 := \begin{pmatrix} 1 \\ 1 \\ 1 \\ \bot \\ \bot \\ \vdots \\ \bot \end{pmatrix} \tag{6.2}$$

Zum Verständnis des Beispiels betrachte man für je zwei Klassendefinitionen die gemeinsamen, also gleichen Attributwerte. Im Mittel führt die Trainingsphase dann stets die Relevanzänderung dergestalt durch, daß die Bedeutungen der gemeinsamen Merkmalsausprägungen für je zwei Klassen geradezu verschwinden. Offensichtlich ist der zu erzielende Lerneffekt unbefriedigend.

Selbstverständlich läßt sich diese Situation über weitere Klassen übertragen, auch können die Wertebereiche gesteigert werden. Wenn die Trainingsphase beim *Umbewerten* der Einzelrelevanzen im ersten Beispiel einen *Zyklus* durchlaufen hat, der am Ende die zuerst erzielten Effekte wieder zunichte macht, so können wir bei einer Erweiterung der entsprechenden Kenngrößen von *zyklischen Ketten* sprechen. Stets bleibt die Trainingsphase erfolglos.

Interessant ist es, dieses Phänomen aus lernpsychologischer Sicht zu betrachten: Wieso kann es prinzipielle, inhärente Schwierigkeiten geben, die Bedeutung gewisser Merkmale bei vorgelegten Problemen zu ermitteln?

Die Ähnlichkeit von Problemstellungen läßt sich eben *nicht* allein durch die Bedeutung bestimmter Merkmale adäquat repräsentieren!

Wenn beispielsweise ein Schachspieler eine erlittene Niederlage analysiert und die Stellung, bei der sein entscheidender Fehler auftrat, zu charakterisieren versucht[3], so genügt es unter Umständen nicht, bestimmte Merkmale dieses Problems zu bewerten (also zu gewichten), sondern vielmehr kann die genaue *Kombination* gewisser ansonsten anders zu gewichtender Attributwerte die Gesamtbewertung völlig verändern!

Bezweifelt wird also die prinzipielle Güte der *linearen* Verrechnung konkreter Gewichte in der Summation der Einzelrelevanzen. Andere, nichtlineare, vielleicht sogar nicht einmal analytische Funktionen würden womöglich zur Problemlösung in sehr schwierig gelagerten Situationen weiterhelfen.

[2] Natürlich ist eine Terminierung möglich, wenn so wenige Fälle in der Fallbasis vorhanden sind, daß die Bedeutung anderer Attribute statistisch signifikant wird oder der Schwellwert auf 1.0 gesetzt wird ...

[3] Dies ließe sich durch zwei Lösungsklassen modellieren: eine *Gewinn-Klasse* und eine *Verlust-Klasse*. Natürlich ist der Spieler bestrebt, nur solche Stellungen (Problembeschreibungen) durch seinen nächsten Zug zu erreichen, deren Lösungen Elemente der Gewinn-Klasse sind

Damit wird jedoch keineswegs die psychologische Adäquatheit des Ähnlichkeitsmaßes generell in Frage gestellt. Vielmehr scheint der menschliche Lerner, wie im Falle des Schachspielers, hin und wieder vor sehr schwierige Probleme gestellt, deren Merkmalsbewertung de facto falsch verläuft. Vielleicht besteht sogar ein Mangel bei der Lösung einiger, für den Menschen zu schwer erscheinender Probleme *auch* darin, daß er aufgrund seiner physiologischen und neuronalen Grundlagen prinzipiell gewichtete Problemmerkmale linear aufaddiert. Allerdings lassen sich die erwünschten Lerneffekte - mit einem erheblichen Mehraufwand - dennoch auch linear erzielen: indem bestimmte Kombinationen von Merkmalen als eigenständige, neue Attribute betrachtet werden ...

Für die vorliegenden Untersuchungen genügt jedoch die Erkenntnis, daß der fallbasierte Ansatz mit dem Ähnlichkeitsmaß (4.9) prinzipielle Beschränkungen aufweist, die im Rahmen der Testumgebung - und im allgemeinen auch in praktischen CBL-Systemen - nicht lösbar sind.

6.1.2 Komplexität der Klassendefinition

Gerade die recht ernüchternde Erkenntnis aus Abschnitt 6.1.1, daß gewisse Problemstellungen schlichtweg zu *komplex* sind, um durch CBL-Systeme effizient gelöst werden zu können, führt zur Einteilung der Klassendefinitionen in *Komplexitätsgrade*, und es wird sich sogar zeigen, daß sich genau diese Unterscheidung in den Resultaten der Testreihe widerspiegelt. Zu beachten ist, daß in diesem Abschnitt stets vorausgesetzt wird, daß keine Klassenüberschneidungen (siehe 6.1.3) auftreten!

Was sind die denkbar einfachsten Klassendefinitionen? Betrachten wir hierzu die beiden Klassen $\mathcal{K}_1$ und $\mathcal{K}_2$:

$$\mathcal{K}_1 := \begin{pmatrix} 1 \\ 2 \\ 1 \\ \perp \\ \perp \\ \vdots \\ \perp \end{pmatrix} \qquad \mathcal{K}_2 := \begin{pmatrix} 2 \\ 2 \\ \perp \\ 1 \\ \perp \\ \vdots \\ \perp \end{pmatrix} \tag{6.3}$$

Die Verhältnisse sind klar. Im ersten Attribut unterscheiden sich die beiden belegten Werte, während im zweiten Attribut Gleichheit auftritt. Gemessen an der Ähnlichkeit der Werte sprechen wir auch von *Spitzen* und *Löchern*. Zum Vergleich hierzu die beiden folgenden Positionen. Zusätzliche und unbekannte Merkmale stellen keine Spitzen und keine Löcher dar, weil es keinen offensichtlichen Grund für eine bestimmte Ähnlichkeitszuweisung gibt. Innerhalb der Klassenelemente treten sowohl Unterschiede als auch Übereinstimmungen in den Einzelvergleichen auf.

Das sechste Merkmal kann ebenfalls weder mit einer hohen noch einer sehr geringen Ähnlichkeit versehen werden, selbst wenn hierbei dasselbe Symbol verwendet wird. Wir nennen es im folgenden auch ein *triviales Merkmal*.

Den obigen Typ von Klassendefinition darf man als *einfach* bezeichnen, weil jeweils nur eine Spitze und ein Loch auftritt. Als *komplex* wollen wir dementsprechend solche Konfigurationen nennen, bei denen mehrere Spitzen und Löcher auftreten. Es wird sich

zeigen, daß einige Lernstrategien auf einfache Klassendefinitionen anders reagieren als auf komplexe. (Vergleiche 7.2.1 bis 7.2.4).

Was aber ist mit dem Beispiel (6.1)? Aufgrund der dritten Klasse können wir nicht mehr von eindeutigen Spitzen und Löchern sprechen, denn - bezogen auf $\mathcal{K}_2$ - stellt schon das erste Attribut sowohl eine Spitze (im Vergleich mit $\mathcal{K}_1$) als auch ein Loch dar (im Vergleich mit $\mathcal{K}_3$).

Wir wollen ein Attribut eine *starke* Spitze nennen, wenn der paarweise Vergleich aller Klassendefinitionen stets eine Spitze ergibt. Dagegen sei eine *schwache* Spitze ein Attribut, das im paarweisen Vergleich mindestens einmal eine Spitze, aber niemals ein Loch darstellt. Analog sei die Terminologie für Löcher festgesetzt.

Ein *gemischtes* Attribut stellt für einige Klassenvergleiche eine Spitze, für andere jedoch ein Loch dar.

Eine Klassenkonfiguration sei *klar*, wenn mindestens eine starke Spitze und keine gemischten Attribute auftreten. Dagegen bezeichnen wir eine Situation wie in den Beispielen (6.1) oder (6.2), in der mindestens zwei gemischte Attribute auftreten und weder Spitzen noch schwache Löcher als *paradox*. Alle anderen Konfigurationen seien *unklar*. Auch hier steigt mit dem Auftreten von Spitzen und Löchern die Komplexität an.

Offensichtlich sind vor allem klare und einfache Klassendefinitionen für den CBL-Einsatz am besten geeignet, während paradoxe Konfigurationen mit mehreren trivialen Attributen[4] überhaupt keinen Lernvorgang erlauben[5]. Wie sich die Komplexität und Klarheit einer Klassendefinitionen auf die einzelnen Strategien auswirken, wird in 7.2.1 bis 7.2.2 eingehend erläutert.

6.1.3 Klassenüberschneidungen

In 6.1.1 wurde aufgezeigt, daß die Terminierung der Trainingsphase an bestimmten Kombinationen von Merkmalen bei der Klassendefinition scheitern kann. Andererseits ist unberücksichtigt geblieben, inwiefern *Klassenüberschneidungen* hierbei eine Rolle spielen. In der Tat können die Testresultate stark verbessert werden, wenn weitere, ähnlich definierte Klassen existieren, wodurch Einzelelemente mehrfache Zugehörigkeiten aufweisen.

Zwar wird in der Trainingsphase jeder Fall bezüglich aller seiner Klassen untersucht und die Terminierung der Relevanzänderung bleibt gefährdet, dennoch können in dieser Situation unter Umständen erstaunliche Ergebnisse bei der Lösung vorgeführter Testprobleme erzielt werden, weil nur *eine* Lösungsklasse ermittelt wird. Diese mag dann gerade derjenigen ohne Zyklen und zyklischen Ketten entsprechen. Gewissermaßen bleibt die Problemlösung erfolgreich, weil es überhaupt nicht mehr der schwer zu handhabenden Klasse bedarf, um ein erwünschtes Ergebnis zu erzielen.

Dieser Effekt könnte durch die Ermittlung *aller* Lösungen eines Problems abgeschwächt werden. Mehr hierzu im Ausblick (Kapitel 8).

[4] Dies ist wichtig, weil anderenfalls möglicherweise die zugehörigen Klassen nur aus einem Element bestünden und somit Zyklen und zyklische Ketten verhinderten

[5] Allein die Terminierung der Trainingsphase genügt nicht als Lernerfolg: Durch den Schwellwert von 1.0 ist dies stets zu erreichen!

6.2 Globales Lernverhalten

Im allgemeinen kann die trainierte Relevanzmatrix als das Produkt eines Lernvorgangs interpretiert werden, der die Bedeutungen der einzelnen Merkmale in jeder Lösungsklasse ermittelt.

Durch den Schwellwert wird das Lernpensum festgelegt. Ein relativ hoher Wert korrespondiert dabei mit einem geringen Lernvorgang und umgekehrt. Allerdings darf - wie mehrfach angeklungen - ein bestimmter, von der jeweiligen Klassendefinition abhängiger Betrag nicht unterschritten werden, weil die Trainingsphase ansonsten nicht mehr terminiert. Die verschiedenen Lernstrategien reagieren unterschiedlich auf diese Situation und in den beiden folgenden Kapiteln werden Lösungsmöglichkeiten geboten, um dieser Schwierigkeit zu entgehen.

Allerdings hängt der Lernerfolg auch von der Präsentation der Trainingsfälle ab. Im bco-Fall zeigt sich ein besonders erfreuliches Bild, aber für sehr geringe Trainingsanteile fehlt genügend Information, um den Lernprozeß erfolgreich abzuschließen. Die Terminierung geschieht hier zu früh. Doch auch für diese Situation zeichnen sich Lösungswege ab, wie in 7.2.3 belegt werden wird.

In 6.1.2 wurde aufgezeigt, daß paradoxe Klassenkonfigurationen nur dann problematisch werden, wenn es zu Zyklen oder zyklischen Ketten kommt, die mehrerer trivialer Attribute bedürfen. In der Praxis kann dies leicht für zumindest eine Teilmenge aller Klassen geschehen, wenn sehr viele Merkmale berücksichtigt werden, obgleich die Probleme durch einige wenige Attribute beschrieben werden können[6].

6.3 Sonstige Aspekte

6.3.1 Innere Zyklen

Sehr viel subtiler als die auftretenden Zyklen in den Klassendefinitionen (vgl. 6.1.1) sind ähnliche Phänomene innerhalb gewisser Einzelfälle. Wenn sie nicht bereits in der Klassendefinition offensichtlich sind, werden sie auch als *innere Zyklen* bezeichnet.

Als Beispiel sei folgende Situation angeführt, die zum Scheitern der Trainingsphase geführt hat, obgleich keine gemischten Attribute in der Klassendefinition auftreten:

Als Klassen seien gegeben:

$$\mathcal{K}_1 := \begin{pmatrix} 3 \\ \perp \\ 2 \\ 2 \end{pmatrix} \qquad \mathcal{K}_2 := \begin{pmatrix} 1 \\ 1 \\ \perp \\ \perp \end{pmatrix} \qquad \mathcal{K}_3 := \begin{pmatrix} \perp \\ 2 \\ 2 \\ \perp \end{pmatrix}$$

Diese Definitionen weisen keine Zyklen und zyklischen Ketten auf und ein flüchtiger Blick könnte zur Annahme verleiten, daß die Gewichte der Merkmale in der Traningsphase schließlich zu stabilen Werten gelangen sollten.

Betrachten wir hierzu folgende drei Fälle:

[6] Eine andere Möglichkeit ist die hohe Gewichtung unbekannter und zusätzlicher Merkmale, weil hierdurch triviale Attribute indirekt überproportional stark gewichtet werden

$$\mathcal{F}_1 := \begin{pmatrix} 3 \\ 1 \\ 2 \\ 2 \end{pmatrix} \qquad \mathcal{F}_2 := \begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \end{pmatrix} \qquad \mathcal{F}_3 := \begin{pmatrix} 1 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

Jeder Fallindex korrespondiert mit der zugehörigen Klasse. Der Zyklus von $\mathcal{F}_2$ in den beiden ersten Attributwerten, bei dem die Gewichtsveränderung im Vergleich zu $\mathcal{F}_1$ gerade wieder aufgehoben wird, wenn der Vergleich mit $\mathcal{F}_3$ stattfindet[7], wird nun allerdings augenscheinlich.

Besonders bemerkenswert ist die Tatsache, daß starke Löcher keineswegs die Wirkung gemischter Attribute aufheben können, sondern diese geradezu fördern. Grundsätzlich bedingen starke Löcher stets Relevanz-Verlagerungen zu den anderen Attributen, sehr viel stärker noch als triviale Attribute dies vermögen!

Die soeben dargelegten Effekte beeinflussen die Testergebnisse im nachfolgenden Kapitel und verringern somit die Unterschiede in den untersuchten Testkriterien, insbesondere was die Qualität der Lernstrategie betrifft.

6.3.2 Stetige Ergänzung der Attribut-Ähnlichkeit

Bei der Betrachtung des Ähnlichkeitsmaßes (4.9) fällt auf, daß eine explizite Unterscheidung zwischen erfüllten und widersprüchlichen Merkmalen durchgeführt wird, obgleich die Grenze θ, bei der eine Attributähnlichkeit als nicht mehr gegeben angesehen wird, willkürlich festgelegt worden ist.

Warum sollte man zwei Attribute mit der Ähnlichkeit $\theta - \varepsilon$[8] als *verschieden* (widersprüchlich) bewerten, während dazu im Gegensatz ein anderes Paar mit der Ähnlichkeit $\theta + \varepsilon$ noch als *gleich* (erfüllt) anzusehen ist?

Anders formuliert stellt sich die Frage, wieso nicht auf eine explizite Trennung zwischen widersprüchlichen und erfüllten Attributwerten verzichtet werden kann.

Folgender Ansatz scheint der Intuition viel näher zu kommen als die Formel (4.9): (Der Einfachheit halber bleiben unbekannte und zusätzliche Attribute unberücksichtigt)

$$sim(\mathcal{P}, C_k^j) = \frac{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j})}{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}) + \beta \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} (1 - \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}))} \tag{6.4}$$

Für den klassischen Fall der Attribut-Unabhängigkeit, bei dem $\zeta(a^i_{\mathcal{P}}, a^i_{C_k^j})$ nur Werte aus $\{ 0, 1 \}$ annehmen kann, erweist sich der obige Ansatz als de facto identisch mit (4.9). Allerdings ergeben sich einige Änderungen bezüglich stetig ausgeprägter Attributähnlichkeit.

Hierzu vereinfachen wir die Gleichung (6.4):

[7] ... und ansonsten keine weiteren Änderungen der Relevanzmatrix stattfinden, was im Testeinsatz tatsächlich zutraf!

[8] wobei ε natürlich als sehr kleine, positive Zahl aufzufassen ist!

$$
\begin{aligned}
sim(\mathcal{P}, C_k^j) &= \frac{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j})}{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}) + \beta \sum\limits_{a^i \in \mathcal{M}} \omega_{ij}(1 - \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}))} \\
&= \frac{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j})}{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}) + \beta \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} - \beta \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \cdot \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}))} \\
&= \frac{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j})}{(\alpha - \beta) \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}) + \beta \sum\limits_{a^i \in \mathcal{M}} \omega_{ij}}
\end{aligned}
$$

wegen der Normierungsbedingung (4.3) gilt jedoch: $\sum\limits_{a^i \in \mathcal{M}} \omega_{ij} = 1$, so daß die letzte Gleichung vereinfacht werden kann zu:

$$
sim(\mathcal{P}, C_k^j) = \frac{\alpha \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j})}{(\alpha - \beta) \sum\limits_{a^i \in \mathcal{M}} \omega_{ij} \zeta(a^i_{\mathcal{P}}, a^i_{C_k^j}) + \beta} \tag{6.5}
$$

Oder in Kurzschreibweise:

$$
sim(\mathcal{P}, C_k^j) = \frac{\alpha E}{(\alpha - \beta) \cdot E + \beta} \tag{6.6}
$$

Für die Änderung der Relevanzmatrix ergibt sich analog zu 5.1.4.1:

$$
\begin{aligned}
\delta_j &= \frac{\alpha(E - \Delta_{\mathcal{E}})}{(\alpha - \beta) \cdot (E - \Delta_{\mathcal{E}}) + \beta} \\
\hookrightarrow \qquad \Delta_{\mathcal{E}} &= E - \frac{\delta_j \cdot \beta}{\alpha - \delta_j(\alpha - \beta)}
\end{aligned}
$$

für den einfachen Fall, daß $\alpha = \beta = 1$ ergibt sich:

$$
\Delta_{\mathcal{E}} = E - \delta_j \tag{6.7}
$$

Die Änderung an den Merkmalen ist somit bereits festgelegt, es kann keine beliebige Verteilungsfunktion gewählt werden.

Dies alles mag nicht gerade gravierend erscheinen, zumal das entstandene Ähnlichkeitsmaß (6.7) sehr kompakt ist. Allerdings führt die Reduktion an Freiheitsgraden im vorliegenden Fall zu wesentlichen Einschränkungen.

Hierzu betrachte man das konkrete Beispiel der Relevanzänderung an zwei Fällen $\mathcal{C}_1$ und $\mathcal{C}_2$ mit drei Merkmalen. Als alte, zu verändernde Gewichte seien gegeben:

$\omega_j^A := \begin{pmatrix} 0.4 \\ 0.3 \\ 0.3 \end{pmatrix}$. Die entsprechenden Attribut-Ähnlichkeiten seien: $\zeta := \begin{pmatrix} 0.9 \\ 0.8 \\ 0.7 \end{pmatrix}$.

Gemäß (6.7) ergibt sich eine Gesamtähnlichkeit der beiden Fälle von
$sim(\mathcal{C}_1, \mathcal{C}_2) = 0.81$.

Dieser Wert sei zu hoch und auf den Betrag der Schwelle δ_j von 0.6 zu bringen:

$$\begin{aligned}
0.9 \cdot \omega_1^N + 0.8 \cdot \omega_2^N + 0.7 \cdot \omega_3^N &= 0.6 \qquad (= \delta_j) \\
\text{weiter gilt:} \qquad \omega_1^N + \omega_2^N + \omega_3^N &= 1 \\
\hookrightarrow \qquad \omega_3^N &= 1 - \omega_1^N - \omega_2^N \\
\text{nach Einsetzung ergibt sich} & \\
0.2 \cdot \omega_1^N + 0.1 \cdot \omega_2^N &= -0.1 \\
\hookrightarrow \qquad \omega_2^N &= -(1 + 2 \cdot \omega_1^N)
\end{aligned}$$

Somit wird für ein positives Gewicht ω_1^N der Wert von ω_2^N stets negativ. Außerdem geht damit zwangsläufig auch die Normierungsforderung an die Relevanzen verloren. Ebenfalls wird die Interpretation der Gewichte als Bedeutungen eines Problem-Merkmals fragwürdig: Wie kann ein Attribut eine negative Bedeutung besitzen?

Der Ansatz zur stetigen Ergänzung, so natürlich er auch erscheint, muß aufgrund schwerwiegender Einschränkungen bezüglich der Freiheitsgrade der Relevanzen verworfen werden.

Teil III

Ergebnisse

Kapitel 7

Testergebnisse und Auswertung

Den nachfolgend angeführten Ergebnissen liegen einige hundert Stunden Rechenzeit[1] und eine große Zahl an verschiedenen Klassenkonfigurationen zugrunde, die zum größten Teil per Zufallsgenerator ausgewählt worden sind. Unter Berücksichtigung der systematischen Beschränkungen durch die verwendete Testumgebung dürfen die Gesamtresultate als relevant[2] betrachtet werden.

Der erste Abschnitt widmet sich den durchschnittlichen Gesamtresultaten ohne Differenzierung der Fallbasen-Charaktristika, während Einzeluntersuchungen im darauf folgenden Abschnitt besprochen werden. Anschließend werden weiterführende Erkenntnisse Erwähnung finden und in der Synthese im letzten Abschnitt des Kapitels wird eine *verbesserte* Lern-Prozedur vorgeschlagen.

7.1 Statistisches Gesamtergebnis

Zu Beginn der Testreihen wurde vermutet, daß einige der Verteilungs- und Änderungsstrategien gegenüber den anderen signifikante Verbesserungen zeigen würden. Diese Vermutung stützte sich bezüglich der Verteilungsfunktionen in erster Linie auf die mathematischen Unterschiede. Das normierte Verhalten aller vier zur Untersuchung gebrachten Funktionen (konstant, proportional, polynomial und exponentiell) wird in Abbildung (7.1) veranschaulicht.

Bemerkenswert ist, daß die Exponentialfunktion im zur Diskussion stehenden Bereich zwischen 0 und 1 eine sehr viel *sanftere* Gewichtung mit geringeren Unterschieden vornimmt als die polynomiale, die hier durch eine quadratische Funktion repräsentiert ist.

Dennoch zeigt sich entgegen der Erwartung im statistischen Mittel im wesentlichen eine große Übereinstimmung der Verteilungsfunktionen, wobei lediglich die konstante Funktion etwas schlechter abschneidet[3] (Abbildung (7.2)).

Es lassen sich allerdings sehr viele Gründe für eine derartige Übereinstimmung finden. Als wichtigste wären zu nennen:

[1] ... mit nicht weniger als 50 Millionen Ähnlichkeitsvergleichen!

[2] Also im Mittel auch im praktischen CBL-Einsatz zu erwarten!

[3] Die gegenüber in der Literatur anzutreffende insgesamt schlechtere Klassifizierungsrate ist auf die Vielzahl von Klassen zurückzuführen (bis zu 15), während den meisten bisher veröffentlichten Ergebnissen lediglich 2 Klassen zugrunde liegen

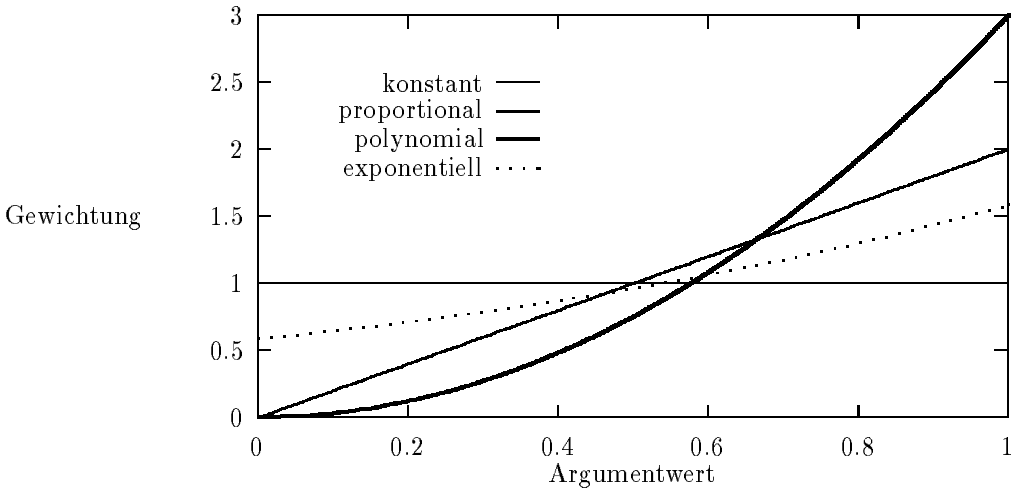

Abbildung 7.1: Verhalten der Verteilungsstrategien

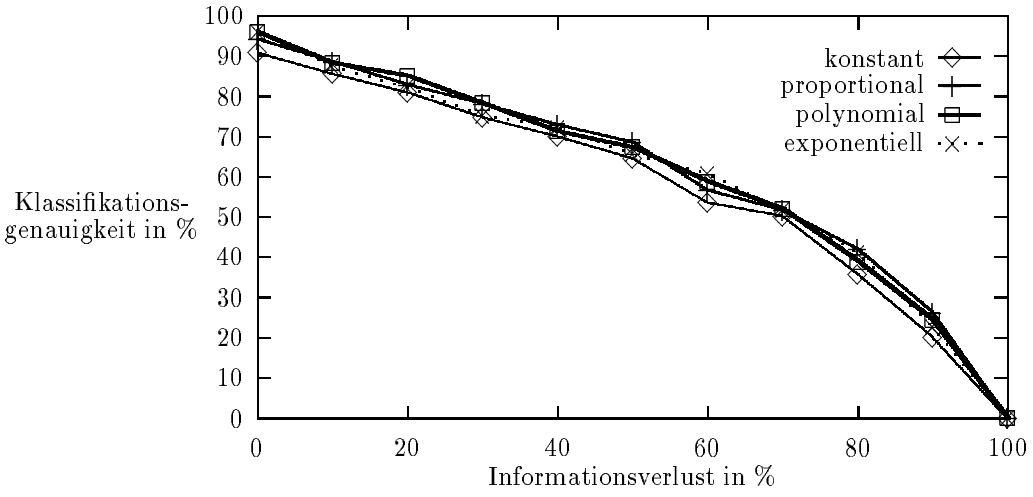

Abbildung 7.2: Lernstrategien bei vollständig gefüllten Fallbasen

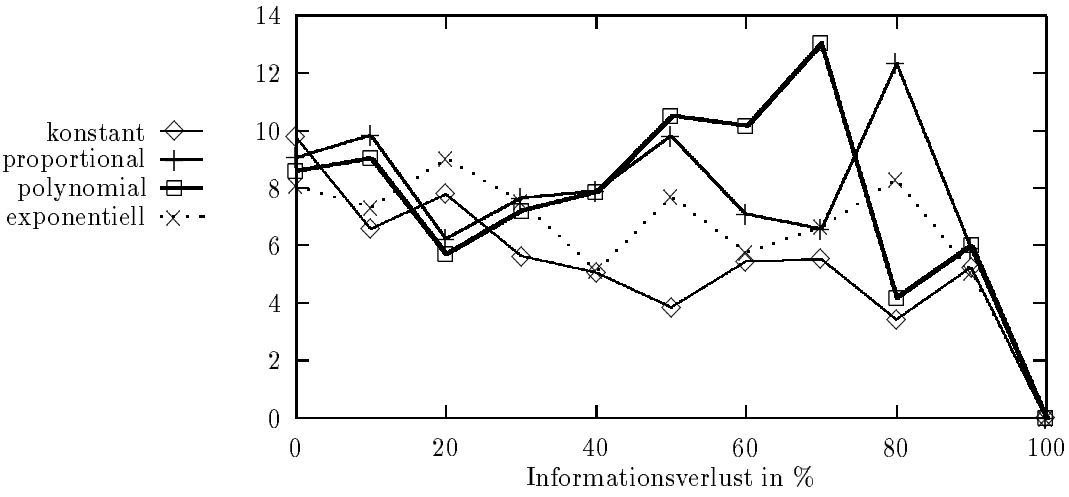

Abbildung 7.3: Standard-Abweichungen der Lernstrategien

- In der Graphik (7.2) sind auch solche Fallbasen berücksichtigt, die keine oder nur sehr wenige Iterationen in der Trainingsphase durchgeführt haben. Dann aber spielt die Verteilungsstrategie für die Klassifikationsfähigkeit eine untergeordnete Rolle.
- Durch die zufällige Konfiguration der Fallbasen finden sich auch solche, die nur wenige Merkmale pro Klassendefinition unterscheiden. Wiederum macht sich die Verteilungsstrategie kaum bemerkbar.
- Klassenüberschneidungen können die Bedeutung der Verteilungsstrategien zunichte machen (vgl. 7.3)
- Vorzüge und Nachteile der verschiedenen Verteilungsstrategien heben sich im statistischen Mittel auf

Besonders der letzte Punkt bedarf der Erläuterung. Die dahinterstehende Behauptung geht dahin, daß die einzelnen Strategien sehr wohl bedeutende Unterschiede zeigen, allerdings nicht für *alle* Typen von Fallbasen, sondern bezogen auf bestimmte Charakteristika. Dies wird in 7.2.4 etwas genauer beschrieben. Als Indiz für eine derartige These sei auf die Standard-Abweichungen in Abbildung (7.3) verwiesen.

Die Situation für die drei verwendeten Änderungsstrategien verdeutlicht die Übersichtsgraphik (7.4).

Wiederum zeigt sich kein eindeutiger Favorit, der für alle Fallbasen unbedingt verwendet werden sollte. Eine genauere Spezifikation über Vorzüge und Nachteile der Änderungsstrategien ist in 7.2.3 enthalten.

Es ist bemerkenswert, daß zwar mit abnehmendem Trainingsanteil im allgemeinen schlechtere Klassifikationsergebnisse erzielt werden, dennoch gilt diese Regel nicht in dem

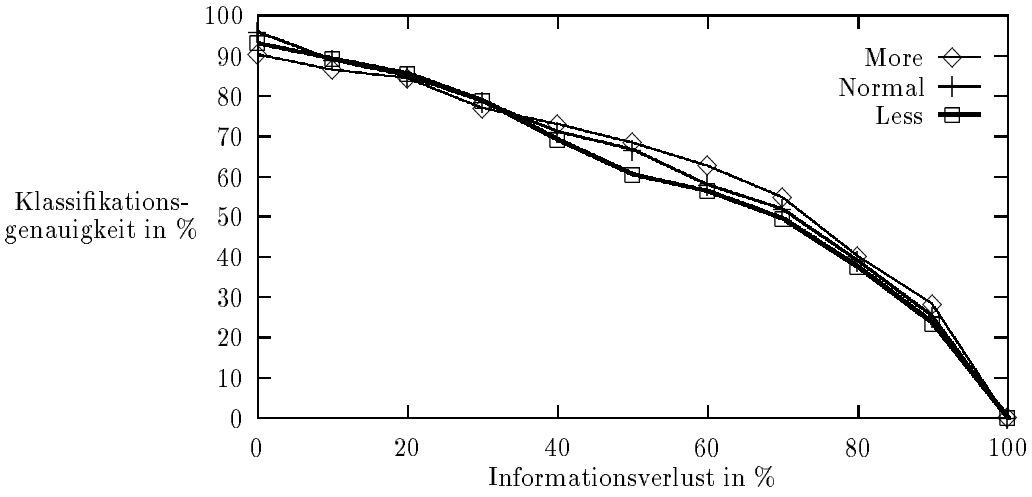

Abbildung 7.4: Änderungsstrategien bei vollständig gefüllten Fallbasen

Maße für Randfälle! Die Abbildung (7.5) zeigt die Änderungsstrategien für diese Art von Fallbasen.

Der Trainingsanteil beträgt hier durchschnittlich knapp 10 bis 30 Prozent und bei zufälliger Auswahl ist ein deutlich schlechteres Ergebnis zu verzeichnen. Lernpsychologisch gedeutet wirkt sich hier der aufbereitete Lerninhalt wesentlich besser auf den Lernerfolg aus als dies im allgemeinen der Fall ist. Der tiefere Grund hierfür ist in der *Säuberung* der Klassen von *redundanten*[4] Fällen zu suchen, der unklare Klassendefinitionen in gewissem Sinne aufhebt.

Das Gesamtergebnis wirkt ernüchternd, weil keine klaren Vorzüge bestimmter Lernstrategien zu verzeichnen sind. Allerdings lassen besonders die Standardabweichungen auf ein deutlicheres Ergebnis bei den Einzelresultaten hoffen. (Weitere dokumentierte Gesamt- und Einzelergebnisse finden sich im Anhang A in tabellarischer Form)

7.2 Fallbasenspezifische Einzelergebnisse

Wie im letzten Abschnitt verdeutlicht, sind die Gesamtresultate der unterschiedlichen Lernstrategien nur unerheblich verschieden. Es wird an dieser Stelle nachgewiesen werden, daß gewisse Typen von Fallbasen dagegen auf bestimmte Ähnlichkeitsmaße und den zugrundegelegten Lernkonzepten sehr unterschiedlich ansprechen.

[4] bezüglich Lernerfolg!

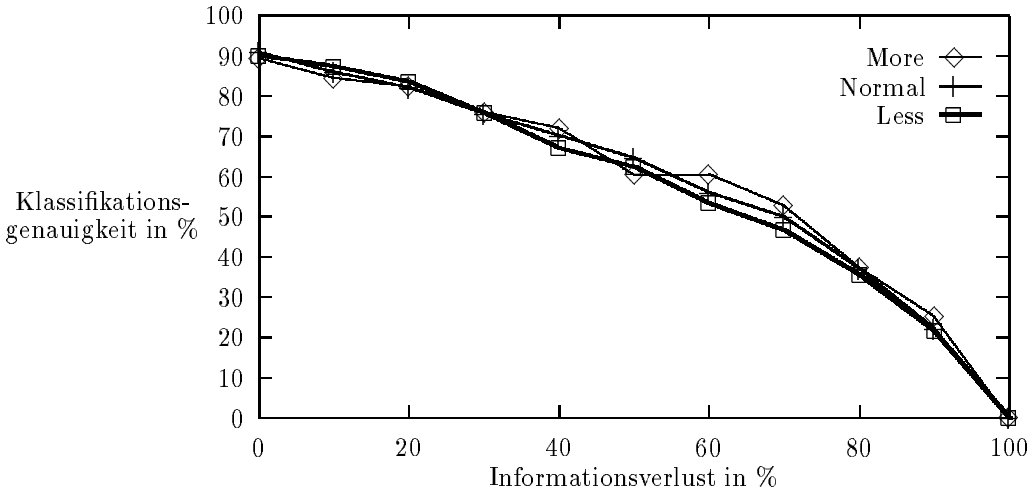

Abbildung 7.5: Änderungsstrategien im bco-Fall

7.2.1 Schwellwert-Belegung

Die wichtigsten *Regulatoren* der Fallbasis, die das Lernvolumen in erheblichem Maße steuern, sind die Klassenschwellwerte der Trainingsphase. Sie sind, wie in 5.1.3 bereits erwähnt, von den Schwellwerten in der Testphase klar zu trennen. Dort stellen sie einen Indikator für die Testkosten dar, während sie hier auf den Lernvorgang in großem Maße Einfluß ausüben.

Sind die Schwellwerte zu hoch angesetzt[5], so wird die Trainingsphase zwar sofort terminieren, aber ohne eine Änderung der Relevanzen! Dagegen sorgt ein zu kleines δ_j quasi immer für eine erfolglose, nicht terminierende Trainingsphase. Den Relevanzen der Matrix wird ein zu großes *Lernpensum* auferlegt. Die Struktur der unterschiedlichen Klassen verhindert das Terminieren. Technisch betrachtet wirken sich zu viele innere Zyklen aus!

Wie sollte man den Schwellwert also anlegen? Konstante Schwellen a priori für alle Klassendefinitionen manuell einzustellen hat sich als gefährlich erwiesen: manche Klassen sind hier überfordert, während für andere dagegen keine Relevanz-Änderung auftritt. Als brauchbarer Richtwert für die Festlegung der Klassenschwelle hat sich der jeweilige Mittelwert der Ähnlichkeiten aller zugehörigen Fälle erwiesen. Nicht selten schwankt dieser Wert zwischen 0.01 und 0.98 und es wird offensichtlich, daß eine fixe Schwelle von etwa 0.7 für keines dieser Extrembeispiele geeignet sein kann. Allerdings sei angemerkt, daß trotz sehr großem Ähnlichkeits-Mittelwert eine gleichhohe Schwelle ein Lernen verhindert. In der Testumgebung wurde deswegen der Kompromiß gewählt, als Klassenschwelle das Minimum zwischen 0.7 und dem Mittelwert zu wählen, was überraschend gute Resultate lieferte.

[5] Als Beispiel verdeutliche man sich die Situation: $\delta_j := 1 \quad \forall j$

Somit wird klar, daß die Anzahl an Trainingsiterationen mit Vorsicht zu genießen ist. Ein zu hoher Wert verhindert die korrekte Terminierung, während die ausbleibende Relevanz-Änderung auch jegliches Lernen verhindert. Außerdem muß die Änderungshäufigkeit mitberücksichtigt werden. Es war zu beobachten, daß der Lernfortschritt mit der Anzahl an Einzeländerungen[6] steigt und nicht etwa mit der Anzahl an Iterationen. Bei gleicher Anzahl an Relevanzänderungen sind also weniger Iterationen zu bevorzugen[7].

7.2.2 Globale Konstanten

Bei den Analysen zu den globalen Konstanten α, β, γ und η hat sich zunächst ein recht unerwarteter Zusammenhang zu den Ähnlichkeitsmittelwerten der einzelnen Klassen ergeben. In Anlehnung an die Bezeichnungen aus 5.2 ergab sich folgende Reihenfolge (bei denselben Klassen!) für die Höhe der Mittelwerte:

$$+\alpha > -\eta > -\beta \geq -\gamma > eq > +\gamma \geq +\beta > +\eta > -\alpha$$

Eine Überbetonung der erfüllten Symptome ($+\alpha$) erzeugt zugleich die größten Ähnlichkeiten der Fälle innerhalb einer Klasse (und natürlich ebenso eine größere Ähnlichkeit der anderen Fälle). Allein dieser Zusammenhang ist kaum beachtenswert, denn es ist klar, daß eine stärkere Betonung der übereinstimmenden oder eine Abschwächung der unterschiedlichen Merkmale stets eine Erhöhung der Ähnlichkeiten mit sich bringt. Viel interessanter ist die *Existenz* einer strikten Reihenfolge[8]! Unter Berücksichtigung dieser Tatsachen lassen sich so die (gesamten!) Klassenmittelwerte steuern. Dies wiederum hat Auswirkungen auf den Schwellwert und damit das Lernpensum. In der Synthese 7.4 werden wir hierauf noch einmal zu sprechen kommen.

In 7.2.1 wurde aufgezeigt, daß die Anzahl an Relevanz-Änderungen ein Indikator für den Lernfortschritt darstellt. Hier zeigte sich folgendes Ergebnis bei klaren Klassendefinitionen:

$$-\beta \gg +\eta > +\alpha \approx -\gamma \approx eq \approx +\gamma \approx -\alpha > -\eta \gg +\beta$$

Besonders auffallend hierzu die Beurteilung der Spitzen und Löcher-Werte in der Relevanz-Matrix. Es handelt sich wiederum um klare Klassendefinitionen, deren Spitzen sehr hoch (≈ 1) und deren Löcher möglichst niedrig (≈ 0) gewichtet werden sollten:

[6] Ausgenommen natürlich das Auftreten von Zyklen und zyklischen Ketten ...

[7] Hier bleibt die Änderungsstrategie unberücksichtigt, die natürlich ebenfalls starke Auswirkungen auf die Anzahl an Relevanz-Verschiebungen hat

[8] Selbst wenn diese Reihenfolge in gänzlich anders gearteten Fallbasen eine andere ist!

$$-\beta \gg +\eta > +\gamma \approx +\alpha \approx eq \approx -\gamma \approx -\alpha > -\eta \gg +\beta$$

Die Besonderheit dieses Resultats besteht ebenfalls weniger in der konkreten Reihenfolge als vielmehr der Tatsache, daß offenbar ein direkter Zusammenhang zwischen den Änderungen der Relevanzmatrix und den erzielten Erfolgen besteht. Bei einfachen und klaren Fallbasen bedeutet quasi jede Relevanzänderung ein Lernfortschritt! Übrigens leitet sich diese Aussage nicht von der Klassifikationsfähigkeit in der Testphase ab - diese war in allen Bereichen etwa gleich und sehr hoch - sondern nur aus der Beurteilung der resultierenden Spitzen- und Löcher-Bewertung. Beispielsweise werden optimale Ergebnisse der Testphase bei Gleichgewichtung aller globalen Konstanten erzielt, wenn nur die aus der Trainingsphase entstandene Relevanzmatrix mit einer Unterbetonung von β hierzu herangezogen wird!

Dies klingt sehr gut für den Fall der klaren Klassendefinition. Aber in der Praxis weiß niemand, welcher Typ von Fallbasis de facto vorliegt, die konkreten Klassendefinitionen stehen hier natürlich nicht zur Verfügung! Außerdem ist die Wahrscheinlichkeit für unklare oder gar paradoxe Definitionen sehr viel größer.

Es zeigt sich, daß der obige Zusammenhang zwischen der Häufigkeit der Relevanzänderung und der resultierenden Spitzen- und Löcherbewertung im unklaren Fall verschwindet! Hier ergab sich als Reihenfolge für die Häufigkeit der Änderungen (im Durchschnitt):

$$-\eta > -\gamma \approx eq \approx +\gamma > +\eta \gg +\alpha > -\beta > +\beta > -\alpha$$

Besonders augenscheinlich wird der Symmetriebruch! Dazu im Vergleich die Güte der Relevanzmatrix:

$$+\alpha \approx -\beta > -\eta \approx -\gamma \approx eq \approx +\gamma \approx +\eta > +\beta \approx -\alpha$$

Ein direkter Zusammenhang kann nicht mehr gefunden werden. Außerdem waren hier die Werte bereits sehr ähnlich, zumindest nicht signifikant verschieden. Offenbar werden hier Änderungen in der Relevanzmatrix durchgeführt, die sich nicht mehr als Lernerfolg interpretieren lassen und vielleicht schon gute Gewichtungen wieder verschlechtern!

Als Resümee über die Verwendung globaler Konstanten kann festgehalten werden, daß für klare Klassendefinitionen ein direkter Zusammenhang zwischen der Anzahl der Änderungen in der Relevanzmatrix und der Güte der erzielten Relevanzen besteht.

Zugleich sprechen starke Unterschiede in der Matrix bezüglich der verschiedenen Konstantengewichtungen für klare Klassendefinitionen[9]!

Alle diese Resultate zeigten sich bei niedrigeren Trainingsanteilen entsprechend auf einem geringeren Niveau, mit gleichen Proportionierungen für die globalen Konstanten!

7.2.3 Änderungsstrategien

Die drei untersuchten Änderungsstrategien, gemäß 5.2 mit *Less*, *Normal* und *More* bezeichnet, bestimmen im wesentlichen die *Lerngeschwindigkeit* des CBL-Systems. Es gibt gute Gründe dafür, die normale Geschwindigkeit zu ändern:

- Bei geringeren Trainingsanteilen ist eine deutlich erhöhte Geschwindigkeit zu empfehlen, damit Spitzen und Löcher genauer detektiert werden können (was jedoch nur bei klaren Klassendefinitionen gänzlich unproblematisch ist)
- Für sehr unklare und paradoxe Fallbasen empfiehlt sich ein geringeres Lerntempo, um die bei auftretenden Zyklen sich gegenseitig auslöschenden Relevanzänderungen zu vermeiden

So hat sich für die Änderungsstrategie *More* gezeigt, daß trotz deutlich geringeren Änderungszahlen bei gleichvielen Iterationen starke Spitzen und Löcher stets deutlicher in der Relevanzmatrix repräsentiert werden. Allerdings gerät diese Strategie besonders bei paradoxen Klassendefinitionen viel schneller in Zyklen und zyklische Ketten, oft bereits nach der zweiten Iteration!

Dagegen scheint *Less* wie geschaffen für paradoxe und unklare Definitionen. Oftmals bleibt die Trainingsphase scheinbar erfolglos, weil keine stabilen Gewichte zu beobachten sind, dennoch waren keine auftretenden Zyklen der Grund: Mit jedem Iterationsschritt verbessert sich die Relevanzmatrix nur wenig, während auftretende Zyklen das Ergebnis sogar verschlechtern!

Aus einer anderen Perspektive betrachtet handelt es sich um das Problem des Lernabbruchs. Bei auch nur halbwegs klaren Verhältnissen erzielt *More* am schnellsten das beste Ergebnis, terminiert dann allerdings noch nicht und verschlechtert die Relevanzmatrix am Ende. Die Abbildung (7.6) zeigt die Leistungskurven der Änderungsstrategien als jeweils nach unten geöffnete Parabeln, deren Optimum im Scheitel zu finden ist. Gegenüber More werden diese Kurven bei Normal und Less deutlich flacher und niedriger, der Scheitelpunkt wird dort jedoch *später*[10] erreicht! *Less* dagegen wird nach einer Reihe von Iterationen bereits abgebrochen, weil der Maximalwert bereits erreicht worden ist, der scheinbar auf Zyklen hinweist. De facto dagegen würde ein weiterer Lernprozeß ein häufig besseres Resultat liefern.

In 7.4 wird eine Möglichkeit aufgezeigt, wie diese Strategien optimal benutzt werden können.

Die *normale* Änderungsstrategie übrigens steht zwischen diesen beiden extremen und erzielt im Mittel aller möglichen Klassendefinitionen sogar das erfreulichste Ergebnis.

[9] Als praktischer Hinweis setzt dies natürlich voraus, daß pro Fallbasis *alle* aus den unterschiedlichen Konstantengewichtungen resultierenden Relevanzmatrizen auch berechnet und verglichen werden. Dies greift ansatzweise die Idee in der Synthese 7.4 vorweg.

[10] Also nach einer größeren Anzahl an Iterationen ...

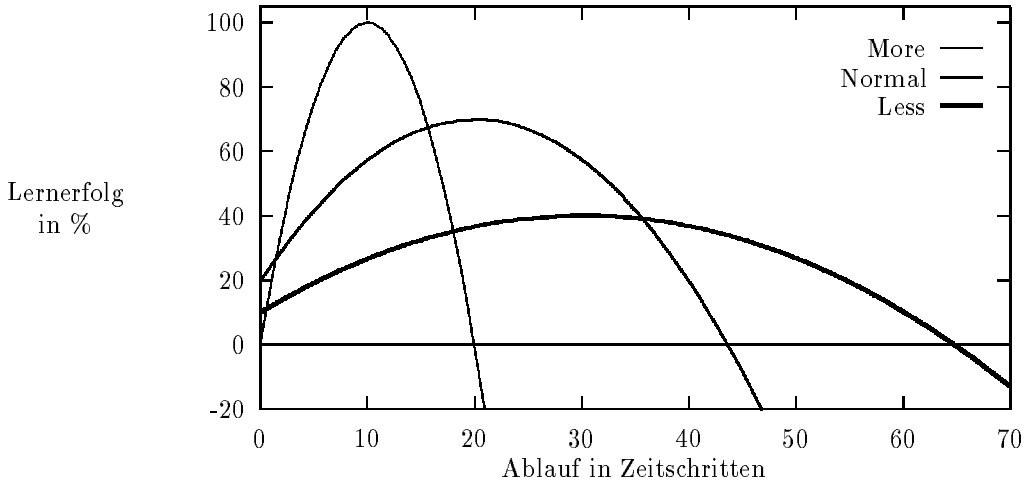

Abbildung 7.6: Leistungskurven

Zur Veranschaulichung dieser Resultate seien exemplarisch die Lernerfolge der drei Strategien an folgendem überschaubaren und typischen Beispiel präsentiert:

$$\mathcal{K}_1 := \begin{pmatrix} \perp \\ \perp \\ \perp \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \end{pmatrix} \qquad \mathcal{K}_2 := \begin{pmatrix} \perp \\ \perp \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \end{pmatrix} \tag{7.1}$$

Die Klassendefinition kann als klar typisiert werden, weil die Attribute 4 und 5 starke Spitzen und 6 bis 8 starke Löcher darstellen, während neben den beiden trivialen Merkmalen (1 und 2) auch das dritte Symptom weder als Spitze noch als Loch zu bezeichnen ist. Als initiale Relevanzmatrix ergibt sich hier:

Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.09523	0.09523	0.09523	0.14285	0.14285	0.14285	0.14285	0.14285
$\mathcal{K}_1$	0.09090	0.09090	0.13636	0.13636	0.13636	0.13636	0.13636	0.13636

Bei einer Standardbelegung der globalen Variablen und Parameter[11] zeigt sich folgender Trainingsverlauf:

[11] $\alpha = \beta = \gamma = \eta = 1$, 100% Trainingsanteil, $\delta_1 = \delta_2 = 0.3$, proportionale Verteilungsstrategie

Bereits nach zwei Iterationen terminieren **Normal** (mit 10 Relevanzänderungen) und **More** (7 Relevanzänderungen), während **Less** dagegen fünf Iterationen mit insgesamt 55 Relevanzänderungen benötigt.

In den nachfolgenden finalen Relevanz-Matrizen sind die Spitzenbewertungen **fett** hervorgehoben:

More - Strategie								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.07436	0.06304	0.07169	**0.36283**	**0.36283**	0.02174	0.02174	0.02174
$\mathcal{K}_2$	0.06846	0.06846	0.07629	**0.36477**	**0.36477**	0.01907	0.01907	0.01907

Normal - Strategie								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.07715	0.06488	0.05819	**0.35000**	**0.35000**	0.03325	0.03325	0.03325
$\mathcal{K}_2$	0.07434	0.06289	0.07120	**0.35000**	**0.35000**	0.03051	0.03051	0.03051

Less - Strategie								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.07603	0.06292	0.04612	**0.34996**	**0.34996**	0.03831	0.03831	0.03831
$\mathcal{K}_2$	0.07517	0.06527	0.06624	**0.34995**	**0.34995**	0.03111	0.03111	0.03111

Die absolute Spitzenbewertung ist hier von geringer Bedeutung, da ein Wert um 0.35 mindestens erforderlich ist, damit die Traningsphase terminieren kann; sein Betrag hängt unmittelbar ab von der Größe der Schwellwerte δ_1 und δ_2. Allerdings erreicht die **More-Strategie** diese magische Grenze viel schneller als die beiden anderen Änderungsstrategien und überschreitet sie meist, was bei klaren Klassendefinitionen kein Fehler sein kann. Diese Kluft wird bei geringeren Trainingsanteilen sogar noch verstärkt.

Für unklare Klassendefinitionen ist jedoch das subtilere Vorgehen der **Less-Strategie** erfolgreicher.

7.2.4 Verteilungsstrategien

Die Wahl der Verteilungsstrategie hat quasi keinen Einfluß auf das Lernvolumen oder die Lernrate, dagegen zeigen sich Unterschiede im *Lernerfolg*. Selbst bei gleichen Iterations- und Relevanz-Änderungszahlen unterscheiden sich die resultierenden Matrizen allerdings nicht unwesentlich voneinander.

Die Graphik (7.1) veranschaulichte bereits die gravierendsten Unterschiede der vier Verteilungsfunktionen. Dennoch folgt ein konkretes Zahlenbeispiel zur Verdeutlichung.

Wenn die alte Gewichtsverteilung beispielsweise eine Summe von

$$\omega^A = 0.4 + 0.3 + 0.2 = 0.9$$

ergäbe und als neues Gewicht der Wert von $\omega^N := 0.6$ gefordert sei. Dann variierten die Verteilungsstrategien folgendermaßen:

Strategie	0.4	0.3	0.2	$\sum = 0.9$
konstant	0.2	0.2	0.2	$\sum = 0.6$
exponentiell	0.220	0.199	0.180	$\sum = 0.6$
proportional	0.267	0.2	0.133	$\sum = 0.6$
polynomial	0.331	0.186	0.082	$\sum = 0.6$

Die konstante und exponentielle Strategie unterstützen jeweils niedrigere Gewichte auf Kosten größerer, während der polynomiale Ansatz die stärksten Gewichte betont. Es ist dabei unvermeidlich und klar, daß nur unwesentliche Unterschiede im Lernvorgang von einfachen Klassendefinitionen zu erwarten sind, wenn die Bewertung starker Spitzen beurteilt werden soll, weil Verteilungsstrategien nur auf mehreren gleichartigen Merkmalen greifen.

Was die Detektion von Löchern angeht, zeigt sich hier eine klare Reihenfolge:

polynomial > proportional > exponentiell > konstant

Dennoch kann selbst die konstante Verteilung sehr nützliche Ergebnisse erbringen, insbesondere für recht unklare Fallbasenkonfigurationen, wenn keine starken Spitzen vorhanden sind und sehr viele Gewichte eine ähnliche Bedeutung erhalten sollten. Dies darf jedoch nicht darüber hinwegtäuschen, daß der Lernprozeß bei konstanten Verteilungsfunktionen einen recht eigenwilligen Verlauf nimmt. Schon der zweite Iterationsschritt kann alle Relevanzänderungen aus dem ersten wieder zunichte machen, was eine sukzessive Verbesserung der Merkmals-Bedeutungen unmöglich macht und die Gefahr, in einen Zyklus zu geraten, stark erhöht. Bei sehr geringem Informationsgehalt in der Fallbasis, wenn ohnehin nur minimale Änderungen erfolgen, kann man die Resultate allerdings durchaus akzeptieren.

Die exponentielle Verteilung steht ein wenig zwischen der natürlichen, proportionalen Strategie und der konstanten. Interessanterweise liefert sie für bestimmte Typen von unklaren Fallbasen einerseits sehr gute, dann aber wieder recht schwache Ergebnisse die darauf zurückzuführen sind, daß Sitzen und Löcher im Schnitt zu ähnlich gewichtet werden. Zur exakten Analyse müßten die Charakteristika der Fallbasen noch wesentlich erweitert werden, um auf konkrete Vorzüge der exponentiellen Funktion hoffen zu können.

Obgleich der polynomiale Ansatz insbesondere bei klaren Klassendefinitionen favorisiert ist, kann seine ständige Anwendung für alle Fallbasen nicht ohne Einschränkungen empfohlen werden, weil beispielsweise die häufig zu beobachtende Relevanzverlagerung von starken Löchern auf schwache bei unklaren Klassendefinitionen unerwünschte Seiteneffekte haben kann und hier der proportionale Ansatz oft ein besseres Ergebnis bringt. Die Gefahr eines Zyklus ist allerdings im polynomialen Fall am geringsten.

Eine mögliche Verwendung auch höhergradiger Polynome mit Koeffizientenadaption wird in der Synthese 7.4 angeboten.

Es ist interessant zu beobachten, wie sich die Verteilungsstrategien im Beispiel aus Abschnitt (7.2.3) darstellen.

Allen vier Strategien genügen zwei Iterationsschritte, wobei die polynomiale 9 Änderungen, die proportionale, wie gesehen, 10, die exponentielle 11 und die konstante Verteilungsstrategie 12 Relevanzänderungen vornimmt[12]. Die Spitzenbewertung ist jeweils gleich, allerdings zeigen sich in der Beurteilung der Löcher deutliche Unterschiede, die **fett** dargestellt sind.

konstante Verteilung								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.05000	0.05000	0.05000	0.35000	0.35000	**0.05000**	**0.05000**	**0.05000**
$\mathcal{K}_2$	0.05000	0.05000	0.05000	0.35000	0.35000	**0.05000**	**0.05000**	**0.05000**

Erwartungsgemäß zeigt die konstante Verteilungsstrategie ein eigenartiges Ergebnis, das in diesem Falle allerdings nicht allzu schlecht beurteilt werden kann.

exponentielle Verteilung								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.05012	0.04984	0.05002	0.35000	0.35000	**0.05001**	**0.05001**	**0.05001**
$\mathcal{K}_2$	0.05009	0.04996	0.04998	0.35000	0.35000	**0.04998**	**0.04998**	**0.04998**

proportionale Verteilung								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.07715	0.06488	0.05819	0.35000	0.35000	**0.03325**	**0.03325**	**0.03325**
$\mathcal{K}_2$	0.07434	0.06289	0.07120	0.35000	0.35000	**0.03051**	**0.03051**	**0.03051**

polynomiale Verteilung								
Kl.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7	Attr. 8
$\mathcal{K}_1$	0.09629	0.10623	0.06788	0.35000	0.35000	**0.00986**	**0.00986**	**0.00986**
$\mathcal{K}_2$	0.01378	0.02017	0.26604	0.35000	0.35000	**0.00001**	**0.00001**	**0.00001**

Die Werte der 3 letzten Attribute von $\mathcal{K}_2$ wurden sogar aufgerundet und es ist schwerlich ein besseres Ergebnis vorstellbar. Allerdings gibt die relativ hohe Gewichtung des zweiten, trivialen Attributs von $\mathcal{K}_1$ zu denken. Offenbar werden hier Relevanzen von starken Löchern nicht unbedingt auf die Spitzen, sondern auch auf triviale Merkmale (und schwache Löcher) verlagert.

Insgesamt stellen sich die Unterschiede der Verteilungsstrategien als wesentlich subtiler, aber auch komplexer heraus als dies etwa bei den Änderungsstrategien der Fall ist. Dennoch zeichnen sich auch hier Ansatzmöglichkeiten für die Verwendung von Verteilungsfunktionen in einem verbesserten Lernschema ab.

[12] Jeweils mit der **normalen** Änderungsstrategie

7.3 Sonstige Erkenntnisse

In 7.2 wurden keine Klassenüberschneidungen berücksichtigt, weil diese de facto die Unterschiede in den Testergebnissen noch weiter verwischen. Der Hauptgrund hierfür besteht in der Vereinfachung bei der Behandlung von Fällen mit mehreren Lösungen, weil lediglich *eine* Diagnose gefunden zu werden braucht. Wenn eine Fallbasis beispielsweise aus vier Klassen besteht, und drei davon zu Zyklen oder zyklischen Ketten neigen, so wird als Diagnose eines Problems mit mehreren Lösungen meist die vierte erzielt werden.

Als Indikator für derartige Überschneidungen wurden bereits die *Impossible Requirements* (IR's) erwähnt. Es gibt jedoch einige Anzeichen dafür, daß Klassenüberschneidungen nur in Verbindung mit unklaren Definitionen IR's liefern. Insgesamt zeigt sich keine erkennbare direkte Abhängigkeit der IR's weder von bestimmten Verteilungs- oder Änderungsstrategien, noch von der Setzung globaler Konstanten. Allerdings führt eine Verminderung der Zyklengefahr auch zu einem Absinken des IR-Auftretens, wodurch die unterschiedlichen Lernstrategien dennoch die IR's beeinflussen können.

7.4 Synthese

In den vorangegangenen Abschnitten wurden die Vorzüge und Nachteile der unterschiedlichen Lernstrategien und Konstantensetzungen in Abhängigkeit vom Typ der Fallbasis analysiert.

Keine der Strategien liefert stets gute Resultate. Vielmehr kann erst eine kombinierte *Selektion* gewisser Strategien ein in diesem Sinne *optimales* Ergebnis produzieren.

Als Idealweg scheint folgendes Vorgehen gelten zu können: In einem ersten Schritt werden durch bestimmte Untersuchungen die Klassencharakteristika herausgefiltert. Verschiedene Heuristiken, wie sie bereits in 7.2 angeklungen sind, dienen sodann dazu, die bestmögliche Relevanzmatrix zu berechnen. Schließlich werden in der Testphase die passenden globalen Konstanten verwendet.

Diese Vision ist allerdings zu optimistisch. Wie bereits im Zusammenhang mit den Verteilungsstrategien (7.2.4) zur Sprache gekommen, müßte die Typisierung der Fallbasen sehr stark verfeinert und entsprechende Heuristiken gefunden werden. Außerdem bleibt stets fraglich, ob allein mit den oben angeführten Mitteln die einzelnen Klassencharakteristika tatsächlich zu extrahieren sind. Darüber hinaus ist der Aufwand zur Typisierung der Fallbasis enorm und in der Praxis kaum realistisch durchführbar.

Es besteht allerdings die Möglichkeit, auch ohne Fallbasentypisierung bessere Ergebnisse zu erzielen und zwar mit folgendem Schema:

1. Berechnung der Mittelwerte $\bar{X}_j$ für die Fall-Ähnlichkeiten einer jeden Klasse
2. Umgewichtung der globalen Konstanten für die Dauer der Trainingsphase, so daß die Mittelwerte $\bar{X}_j$ eine nach den Dimensionen der Fallbasis ausgerichtete Konstante ψ_1 (z.B: $\psi_1 := 0.5$) überschreiten
3. Setzen der Schwellwerte $\delta_j := min\{\bar{X}_j, \psi_2\}$ (wobei gilt: $\psi_1 < \psi_2 < 1$)
4. Parallele Durchführung der Trainingsphase mit den drei Änderungsstrategien und proportionaler Verteilungsstrategie, oder anstatt des proportionalen Ansatzes eventuell eine polynomiale Strategie mit adaptiven Koeffizienten
5. Nach Vergleich der Ausgangs- und den drei Ergebnis-Relevanzen Auswahl der besten Werte (gemäß Tabelle 7.1) bzw. ebenfalls Adaption der Polynom-Koeffizienten[a]
6. Durchführung der Testphase mit den ursprünglichen globalen Konstanten und testkostenspezifischen Schwellwerten

[a] falls im Punkt 4 eine polynomiale Verteilungsstrategie gewählt worden ist

Als Heuristik für die Wahl der besten Relevanzmatrix können die klassenspezifischen *Veränderungen* gegenüber den Ausgangsrelevanzen dienen! Hierzu sei in Erinnerung gerufen, daß *Less* gegenüber *Normal* und *More* für gewöhnlich die langsamste Lernrate aufweist. Demgemäß sollten die Veränderungen bei More auch am größten sein[13]. Wenn dem nicht so ist, hat diese Strategie ihr Optimum bereits überschritten und die erzielten Relevanzen sind höchstwahrscheinlich wertlos. Daß diese Situation auf unklare Klassendefinitionen hinweist, braucht nun aber - im Gegensatz zur *Idealprozedur* - nicht berücksichtigt zu werden.

Von Bedeutung ist also die *Reihenfolge* der Veränderungsgrößen! Idealerweise sollte diese Reihenfolge für jede Einzelrelevanz berechnet werden, vermutlich genügt jedoch die Betrachtung der Summe aller positiven Relevanzänderungen einer Klasse, sinnvollerweise bezogen auf die *Less*-Strategie[14].

In der Tabelle 7.1 sind alle möglichen Kombinationen von Reihenfolgen für die Größe der Relevanz-Veränderungen der einzelnen Strategien aufgeführt. Ebenfalls ist dargestellt,

[13] Allerdings müssen hier, insbesondere bei annähernd gleichen Resultaten, die absoluten Beträge auf die Anzahl an Relevanzänderungen relativiert werden!

[14] Was aber, wenn auch *Less* bereits ihr Optimum überschritten hat? Dies dürfte bei einer grundsätzlichen Beschränkung der Iterationen auf etwa 20 - je nach Dimensionen der Fallbasis - nur sehr selten vorkommen ...

Reihenfolge	**Less**	**Normal**	**More**	**Koeff.-Adaption**
o < l < n < m	pessimistisch	Kompromiß	optimistisch	nein
o < l < m < n	pessimistisch	optimistisch	fehlerbehaftet	nein
o < n < l < m	optimistisch	fehlerbehaftet	fehlerbehaftet	nein
o < n < m < l	optimistisch	fehlerbehaftet	fehlerbehaftet	nein
o < m < n < l	optimistisch	fehlerbehaftet	fehlerbehaftet	nein
o < m < l < n	pessimistisch	optimistisch	fehlerbehaftet	nein
n < o < l < m	optimistisch	fehlerbehaftet	fehlerbehaftet	ja
n < o < m < l	optimistisch	fehlerbehaftet	fehlerbehaftet	ja
m < o < l < n	pessimistisch	optimistisch	fehlerbehaftet	nein
m < o < n < l	optimistisch	fehlerbehaftet	fehlerbehaftet	nein
m < n < o < l	optimistisch	fehlerbehaftet	fehlerbehaftet	eventuell
n < m < o < l	optimistisch	fehlerbehaftet	fehlerbehaftet	ja

(*o : original Relevanzen, l, n, m : Ergebnis bei Less, Normal und More*)

Tabelle 7.1: Relevanz-Auswahl

welche dieser Möglichkeiten für die Ergebnis-Relevanzmatrix herangezogen werden sollte. Bei mehreren zur Verfügung stehenden Auswahlmöglichkeiten wird zwischen einer optimistischen und einer pessimistischen Wahl unterschieden. So könnten beispielsweise die Werte für *More* bereits überzogen sein, obwohl die Reihenfolge *Less* < *Normal* < *More* noch[15] zutrifft. Das *More*-Ergebnis zu wählen wäre dann optimistisch, das *Less*-Resultat dagegen pessimistisch.

Die letzte Spalte weist darauf hin, ob der gesamte Lernvorgang unbefriedigend abgelaufen ist. Wenn dies zutrifft, bietet es sich im Falle einer polynomialen Verteilungsstrategie an, nach einer leichten Abänderungen der Koeffizienten[16] den Lernschritt mit der Ausgangsmatrix zu wiederholen; eventuell solange, bis ein erfreulicheres Ergebnis erzielt wird...

Insgesamt ist mit einer drastischen Ergebnisverbesserung unter Verwendung des oben dargestellten Schemas zu rechnen, weil gegenüber dem bisherigen Vorgehen (CBL4) die Lernrate durch parallele Berechnung *kontrolliert* und adaptiert wird! Die endgültige Rechtfertigung des entstandenen Mehraufwandes kann sich dagegen erst im praktischen Einsatz zeigen. Dabei ist zu bedenken, daß keinesfalls Fall eine Klassifizierungs-Verschlechterung zu erwarten ist, da im Zweifelsfalle, insbesondere bei einer falschen Reihenfolge der Änderungsstrategien, einerseits ohnehin wieder auf den normalen Ansatz zurückgegriffen werden kann, während andererseits nun ein Indikator für einen besonders schwierigen Lernprozeß zur Verfügung steht...

[15] Vielleicht wäre bereits im nächsten Iterationsschritt die Reihenfolge eine andere!?

[16] Wie und in welchem Maße bedarf der eingehenden Analysen ...

Kapitel 8

Zusammenfassung und Ausblick

Aus der Vielzahl von möglichen Untersuchungen über die dynamischen Lernregeln in CBL-Systemen wurden sowohl die Verteilungs- und Änderungsstrategien, als auch die globalen Konstanten und Klassenschwellwerte ausgewählt.

Die Änderungsstrategien sind dabei für das *Maß* der Relevanz-Änderungen verantwortlich und die Verteilungsstrategien für die genaue Gewichtung der erfüllten, widersprüchlichen und unbekannten Merkmale.

Im Durchschnitt aller möglichen Fallbasen erweist sich keine dieser Lernstrategien als besonders empfehlenswert. Dagegen werden die Unterschiede in Abhängigkeit vom *Typ der Klassendefinition* offensichtlich.

Als denkbares Resultat dieser Erkenntnis wurde in 7.4 ein Schema entwickelt, das die fallbasenspezifischen Charakteristika berücksichtigt, ohne sie explizit herauszufiltern, indem *parallel* die Ergebnisse der Änderungsstrategien berechnet und anschließend heuristisch verglichen werden. Bei einem solchen Vorgehen sind deutlich bessere Lernerfolge für Case-Based Learning Systeme zu erwarten.

Erweiterbar bleibt dieses System durch Hinzunahme *aller Lösungen* eines Problems bei Klassenüberschneidungen. Hierzu ist lediglich die Testphase dahingehend zu ergänzen, anstatt die Diagnose mit der größten Fallähnlichkeit allein als Lösung anzusehen, nunmehr *alle* Diagnosen mit Fallähnlichkeiten über dem Schwellwert δ_j (der Testphase) als Lösungen des Problems zu verwenden. Allerdings wird hier die Verifikation problematisch, wenn nur ein Teil der korrekten Lösungen vom CBL-System erkannt wird.

Im übrigen könnte die Berechnung des wichtigsten Lern-Regulators, nämlich der Trainingsschwelle δ_j, deutlich subtiler verlaufen. Ein Adaptionsvorgang bietet sich hier an, der vermöge der Anzahl an Relevanzänderungen gesteuert wird: Wenn zuwenige Änderungen in der Trainingsphase geschehen sind, könnte der Schwellwert gesenkt werden, während hingegen im umgekehrten Fall, beim Auftreten von Zyklen, sukzessive dieser Wert zu erhöhen ist. Wer die Änderungen des Ähnlichkeitsmaßes bereits als einen Meta-Lernvorgang in CBL-Systemen betrachtet, müßte die Adaption des Schwellwertes als *Meta-Meta-Lernprozeß* verstehen, weil das Meta-Lernen vermöge Änderung der Relevanzmatrix selbst wiederum durch die Schwellwertanpassung gesteuert, oder auch gelernt wird.

Schließlich sei auf den wichtigsten Grund verwiesen, weshalb komplexe Ähnlichkeitsmaße wie (4.9) hier statistischen *Polynomklassifikatoren* (siehe z.B. [Mol92] oder [Sch77])

für CBL-Systeme vorzuziehen sind. Zwar mögen statistische Verfahren sehr gute Resultate bei der Klassifikation von Problemen liefern, ihr Vorteil, die Einfachheit nämlich, bedeutet aber zugleich einen schweren Nachteil: es lassen sich kaum Parameter einstellen, die den Lernvorgang entscheidend beeinflussen könnten. Optimistisches oder pessimistisches Lernen läßt sich nur schwerlich modellieren. Wie sollte hier auf spezifische Eigenheiten von Fallbasencharakteristika eingegangen werden? Deshalb kann auf den lernpsychologisch orientierten Ansatz (4.9) nicht verzichtet werden ...

Anhang A

Tabellarische Resultate

Die graphisch dargestellten Gesamtresultate aus dem 7. Kapitel werden an dieser Stelle zur Verdeutlichung tabellarisch wiedergegeben und ergänzt. Die Werte verstehen sich als Klassifikationserfolg.

Im ersten Abschnitt finden sich die Verteilungsstrategien, während im darauf folgenden Abschnitt die Änderungsstrategien abgehandelt werden.

A.1 Verteilungsstrategien

Inf.-Verlust (%)	konstant	proportional	polynomial	exponentiell
0	90.8	94.1	95.8	95.1
10	85.6	88.6	88.2	87.2
20	81.0	82.8	85.0	82.3
30	74.9	78.0	78.2	75.2
40	70.0	72.8	71.1	72.2
50	64.7	68.6	67.2	65.5
60	53.6	56.6	58.8	60.8
70	50.3	51.6	52.0	51.6
80	35.8	42.0	39.0	41.0
90	20.2	26.1	24.3	22.6
100	0.0	0.0	0.0	0.0

Tabelle A.1: Klassifikationsrate (%) bei einem Trainingsanteil von 100 %

Inf.-Verlust (%)	konstant	proportional	polynomial	exponentiell
0	9.8	9.1	8.6	8.1
10	6.6	9.9	9.0	7.3
20	7.8	6.3	5.7	9.0
30	5.6	7.7	7.2	7.5
40	5.1	8.0	7.9	5.1
50	3.9	9.9	10.5	7.7
60	5.5	7.1	10.2	5.8
70	5.6	6.6	13.0	6.7
80	3.4	12.4	4.18	8.3
90	5.3	6.0	6.0	5.1
100	0.0	0.0	0.0	0.0

Tabelle A.2: Standardabweichung bei einem Trainingsanteil von 100 %

Inf.-Verlust (%)	konstant	proportional	polynomial	exponentiell
0	88.8	92.1	92.8	91.4
10	83.4	86.2	86.8	86.0
20	78.3	80.1	83.1	79.4
30	72.4	75.4	76.0	74.5
40	68.2	70.6	70.7	68.6
50	59.1	65.0	61.2	63.3
60	51.4	54.6	60.1	58.7
70	43.8	50.8	50.3	49.6
80	32.6	36.1	38.2	39.6
90	22.4	25.0	26.2	20.6
100	0.0	0.0	0.0	0.0

Tabelle A.3: Klassifikationsrate (%) bei einem Trainingsanteil von 50 %

Inf.-Verlust (%)	konstant	proportional	polynomial	exponentiell
0	12.3	8.3	8.7	12.1
10	10.2	8.2	7.5	9.3
20	9.9	12.4	7.0	7.4
30	7.1	10.0	11.31	6.9
40	11.1	6.1	10.5	4.9
50	11.8	11.1	9.1	11.4
60	14.6	7.4	10.9	13.7
70	14.5	8.9	10.5	15.4
80	12.1	4.4	10.0	14.1
90	13.7	11.3	5.8	17.2
100	0.0	0.0	0.0	0.0

Tabelle A.4: Standardabweichung bei einem Trainingsanteil von 50 %

Inf.-Verlust (%)	konstant	proportional	polynomial	exponentiell
0	81.6	85.0	89.0	84.0
10	80.1	83.0	84.8	78.8
20	74.3	77.2	80.4	76.5
30	68.8	73.4	75.2	69.7
40	63.6	68.2	66.1	62.5
50	57.1	63.3	60.7	54.0
60	48.4	52.6	53.2	47.1
70	41.1	46.1	47.8	42.4
80	31.2	33.6	35.4	32.6
90	15.3	21.0	19.2	20.6
100	0.0	0.0	0.0	0.0

Tabelle A.5: Klassifikationsrate (%) bei einem Trainingsanteil von 10 %

Inf.-Verlust (%)	konstant	proportional	polynomial	exponentiell
0	12.1	15.2	11.6	12.1
10	10.9	11.6	10.0	12.2
20	11.3	13.3	11.4	14.2
30	9.7	13.9	12.6	9.1
40	13.4	8.3	12.7	8.7
50	9.6	10.5	9.5	11.6
60	8.0	8.7	8.9	5.3
70	7.7	8.7	10.1	11.5
80	4.7	6.1	7.3	6.7
90	5.6	5.3	3.1	5.2
100	0.0	0.0	0.0	0.0

Tabelle A.6: Standardabweichung bei einem Trainingsanteil von 10 %

Es zeigt sich stets dieselbe Tendenz: Alle Werte der Strategien sind im Mittel nicht signifikant verschieden, obgleich die konstante Verteilung in der Regel die schwächsten Resultate liefert.

A.2 Änderungsstrategien

Die Änderungsstrategien **Less**, **Normal** und **More** weisen ebenfalls nur geringe Unterschiede in den statistischen Mittelwerten auf.

Dennoch läßt sich bereits an den Gesamtresultaten ablesen, daß **More** bei geringerem Trainingsanteil gegenüber den beiden anderen Änderungsstrategien bessere Ergebnisse erzielt.

Inf.-Verlust (%)	Less	Normal	More
0	93.0	96.0	90.3
10	89.2	89.1	86.5
20	84.6	84.1	83.4
30	78.7	78.7	77.0
40	69.0	71.2	73.1
50	61.4	67.7	69.4
60	56.3	58.1	62.6
70	49.5	52.1	54.8
80	37.4	39.1	40.3
90	22.4	23.9	26.8
100	0.0	0.0	0.0

Tabelle A.7: Klassifikationsrate (%) bei einem Trainingsanteil von 100 %

Inf.-Verlust (%)	**Less**	**Normal**	**More**
0	91.1	92.9	89.8
10	86.6	86.8	85.1
20	80.2	80.5	80.2
30	75.4	75.1	76.2
40	67.5	70.1	72.6
50	59.3	65.3	66.7
60	54.2	56.9	59.9
70	48.1	50.5	53.3
80	36.2	37.8	39.7
90	22.7	25.1	27.5
100	0.0	0.0	0.0

Tabelle A.8: Klassifikationsrate (%) bei einem Trainingsanteil von 50 %

Inf.-Verlust (%)	**Less**	**Normal**	**More**
0	85.5	86.7	87.9
10	82.2	83.7	84.1
20	76.5	77.6	77.9
30	72.2	73.0	72.8
40	63.1	66.3	67.7
50	54.2	60.7	62.3
60	50.4	52.3	53.2
70	44.2	45.9	47.5
80	32.1	33.3	35.4
90	18.5	20.7	20.5
100	0.0	0.0	0.0

Tabelle A.9: Klassifikationsrate (%) bei einem Trainingsanteil von 10 %

Trainingsanteil (%)	**Less**	**Normal**	**More**
100	4924	484	492
90	3383	415	446
80	2335	400	315
70	2155	307	269
60	1924	269	261
50	1693	229	223
40	1002	176	200
30	309	107	161
20	231	72	84
10	65	19	9

Tabelle A.10: Durchschnittliche-Änderungsrate bei einem Trainingsanteil von 10 %

Die Tabelle A.10 stellt die durchschnittliche Anzahl an Änderungen der Relevanzmatrix der verschiedenen Änderungsstrategien dar.

Besonders bemerkenswert ist hier, daß **More** gegenüber **Normal** nichts stets geringere Werte zeigt, was auf ein Überschreiten des geforderten Lernpensums (vgl. 7.2.3) schließen läßt.

Anhang B

Implementierung

Die in Kapitel 5 beschriebene Testumgebung wurde in Turbo Pascal [TP91] programmiert und die Daten auf meheren PC's zum Teil gleichzeitig berechnet.

Die folgenden Abschnitte behandeln jeweils die verwendeten Module (*Units*), wobei auf eine detaillierte Erläuterung der Unit *Crt_Refi(ne)* verzichtet wird, weil hier lediglich hardware-spezifische Möglichkeiten für die Ein- und Ausgabe bereitsgestellt werden.

B.1 Übersicht der erstellten Units

Die Abbildung B.1 zeigt die - neben denen des Systems - verwendeten *Units* und deren Zielsetzung.

Alle diese Module stellen Prozeduren und Funktionen bereit, die das Hauptprogramm einbindet und verwendet.

Zunächst erfolgt eine Beschreibung dieser obersten Programmebene, während anschliessend die einzelnen Units behandelt werden. Es findet jeweils eine Auswahl der wesentlichsten Prozeduren und Funktionen statt, die im Quellcode dargestellt und erläutert werden.

Die Kommentare werden dabei *kursiv* hervorgehoben.

1. UNIT CRT_REFI(NE):
   - Prozeduren zur Ergänzung und Erweiterung der verfügbaren Ein- und Ausgabemöglichkeiten
2. UNIT INIT(IALIZATION):
   - Definition der Datenstrukturen sowie Prozeduren zur Initialisierung der benutzten Variablen
3. UNIT BASICS:
   - Grundlegende Prozeduren, die von den anderen Units aufgerufen werden
4. UNIT TRAINING:
   - Prozeduren zur Durchführung der Trainingsphase
5. UNIT TESTING:
   - Prozeduren zur Durchführung der Testphase
6. UNIT AUTOMATI(C RUN):
   - Zusätzliche Prozeduren zur zufälligen Dimensionierung der Fallbasis sowie zur Aufnahme größerer Test-Datenmengen

Abbildung B.1: Die Units im Überblick

B.2 Hauptprogramm

B.2.1 Übersicht

- Verwendete Units:
  - Crt_Refi
  - Init
  - Basics
  - Training
  - Testing
  - Automatic
- Exportierte Funktionen: 0
- Exportierte Prozeduren: 0
- Target Code (in Bytes): 48720
- Data Code (in Bytes) : 62584

B.2.2 Implementierung

```
program cbl_analyse;
begin
   if run_allone_test then
     begin
       repeat
         init_global_variables;
         init_casebase_dimensions;
         define_random_casebase;
         for treshold_calculation := manual to mean_of_weights do
           for global_variable_setting := 0 to 8 do
           +α...eq...−α
             for distribution_function := konstant to exponential do
             konstant, proportional, polynomial, exponential
               for relevance_change := half_change to double_change do
               Less, Normal, More
                 for casebase_filling := 11 downto 0 do
                 100% - 10%, 5%, bco
                   begin
                     set_global_variables (global_variable_setting);
                     set_casebase_filling (casebase_filling);
                     init_relevance_matrix;
                     get_means_of_tresholds (treshold_calculation);
                     store_presettings (testfile_counter);
                     data documentation
                     training_phase;
```

```
                    store_training_results (testfile_counter);
                    for noise := 0 to 10 do
                    0% - 100%
                      begin
                        set_noise (noise);
                        testing_phase (testfile_counter);
                      end;
                    shrink (testfile_counter);
                    reduces data-space
                    inc (testfile_counter);
                  end; { for casebase filling }
      until keypressed;
      calculation until key break
    end { if run_allone_test = true }
  else
    begin
      init_global_variables;
      define_casebase;
      init_relevance_matrix;
      determine_tresholds (tresh_calc_mode);
      set tresholds after means - calculation
      output_casebase_definition (true);
      output_relevance_matrix (true);
      shows results on the screen / write them into a file (=false)
      training_phase;
      output_relevance_matrix (true);
      testing_phase (testfile_counter);
    end; { if run_allone_test = false }
end.
```

B.3 Unit Init

B.3.1 Übersicht

- Verwendete Units:
  - Crt_Refi
- Exportierte Funktionen : 1
- Exportierte Prozeduren : 3

B.3.2 Implementierung

B.3.2.1 Datenstruktur-Deklarationen

```
interface
selection

const
        max_nr_of_symptoms = 15;
        example for highest value of symptom number
        max_attrib_values = 7+1;
        example for highest value of attribute range
        testcase_factor = 1
        nr of times to test all cases
        turn_dc_weight_to_0 = false;
        if this flag is true, all * (don't care) weights are set to zero
        max_iterations = 20;
        number of iterations during the training phase
        first_class = 'A';
        last_class = 'P';
        max_nr_of_classes = 16;
        the number of possible classes is limited
        to 16 to store the class vectors in words (16 bits)
        max_nr_of_cases = 10000;
        arithmetic_minimum = 0.00001;
        limit to avoid arithmetc errors

type
        case_type = array [1..max_nr_of_symptoms] of char;
        distribution_function_types =
             (constant, proportional, polynomial, exponential);
        relevance_change_types =
             (half_change, normal_change, double_change);
        class_vector = word;
        packs class correspondence bitwise
        classname_type = char;
        rel_matrix_type =
             array [first_class..last_class,1..max_nr_of_symptoms]
```

```
            of real;
    type for relevance matrix
    case_base_def_type =
         array [first_class..last_class,1..max_nr_of_symptoms]
              of char;
    treshold_array_type =
         array [first_class..last_class] of double;
    case_list_type =
        record
            case_number : longint;
            classes : class_vector;
        end; { record }
    stores reference of the case number and its classes
    case_list_array = array [1..max_nr_of_cases] of case_list_type;
    treshold_calc_types = (manual, mean_of_weights);
    class_stat_type =
        record
            mean, deviation : real;
            class_counter : word;
        end; { record }
    datastructure for storing means and deviations of all classes
var
    case_base_def : case_base_def_type;
    nr_of_classes : byte;
    nr_of_all_cases,
    nr_of_relevant_cases : longint;
    relevant cases are those who have at least one class
    max_class_name: classname_type;
    alpha, beta, gamma,
    eta, theta,
    distributor_c : double;
    relevance_matrix : rel_matrix_type;
    case_array : case_list_array;
    treshold_array : treshold_array_type;
    iteration_counter : integer;
    bco_base_size : word;
    size of the case base for bco test
    treshold_calculation : treshold_calc_types;
    nr_of_symptoms,
    nr_of_attrib_values : byte;
    distribution_function : distribution_function_types;
    relevance_change : relevance_change_types;
    casebase_filling : byte;
    probability_testcase : real;
    class_statistics :
         array [first_class..last_class] of class_stat_type;
```

```
noise_probability : real;
attrib_dependence : boolean;
hole_changes_counter,
hole_ir_counter,
testfile_counter,
longtime_case_counter,
longtime_class_counter,
longtime_sim_counter : longint;
```

B.3.2.2 Prozedur- und Funktions-Deklarationen

`interface`
selection

`procedure init_global_variables;`
resets all global variables

`function potence (base, exponent : byte) : longint;`
claculates the arithmetic potence function

`procedure define_casebase;`
reads case_definitions from the keyboard

`procedure error (error_nr : byte);`
performs the error handling

B.4 Unit Basics

B.4.1 Übersicht

- Verwendete Units:
  - Crt_Refi
  - Init
- Exportierte Funktionen : 11
- Exportierte Prozeduren : 14

B.4.2 Implementierung

```
interface
```
selection

```
function number_value (ch : char) : byte;
```
transforms the attribute values into numbers ('1' ⇒ 1 unknown ⇒ 0 ...)

```
function char_value (b : byte) : char;
```
transforms numbers into attribute values (1 ⇒ '1'; 0 ⇒ unknown ...)

```
function case_attribute_similarity
      (comp_attribute, situation_attribute : char) : double;
```
returns the similarity [0..1], u, z of both case attrbiutes

```
var
     actual_case_val,
     comp_case_val : byte;
     result : double;

begin
   if (comp_attribute = unknown) and
     (situation_attribute = unknown) then result := both_not_there
     else if situation_attribute = unknown then result := unknown_value
     else if comp_attribute = unknown then result := redundant_value
     else
       begin
         comp_case_val := number_value (comp_attribute);
         actual_case_val := number_value (situation_attribute);
         if attrib_dependence then
           begin
             result := 1.0 - abs ((comp_case_val - actual_case_val)
                       / nr_of_attrib_values);
             if result < arithmetic_minimum then result := 0.0;
           end
         else if comp_case_val = actual_case_val
                 then result := 1.0
                 else result := 0.0;
```

```
      end;
   case_attribute_similarity := result
end;

function lowest_class (classes : class_vector) : classname_type;
```
returns the classname of the first class form the class vector

```
function attribute_similarity
   (case_attrib, class_attrib : char) : double;
```
the attribute similarity is calculated as difference

```
function high_similarity
   (case_attrib, class_attrib : char) : boolean;
```
true, if both attribute values are very similar (dep. on theta)

```
begin
   high_similarity :=
     attribute_similarity (case_attrib, class_attrib) >= theta;
end;

function determine_case_nr (c : case_type) : longint;
```
calculates the nr of the case by multiplying the sym-values

```
procedure generate_case_values (var c : case_type; nr : longint);
```
calculates the case values by dividing the case number

```
function determine_case_classes (c : case_type) : class_vector;
```
puts all the classes of a case into a small_string

```
function determine_case_classes_of_case_nr
   (nr : longint) : class_vector;
```
puts all the classes of a case number into a small_string

```
function common_classes
   (classes_1, classes_2 : class_vector) : boolean;
```
returns true iff both class vectors have any common classes

```
procedure init_relevance_matrix;
```
sets the values of the relevance matrix according to (4.3.4.1)

```
procedure output_relevance_matrix (screen : boolean);
```
shows the contents of the relevance_matrix on the screen (screen = true) or on the printer

```
procedure output_casebase_definition (screen : boolean);
```
shows the casebase on the screen (screen = true) or on the printer

```
procedure output_tresholds (screen : boolean);
```
shows the treshold_array on the screen (screen = true) or on the printer

```
procedure case_out (case_ex : case_type);
```
shows the case components on the screen

```
procedure class_out (class_ex : class_vector);
```
shows the binary class vector components on the screen

```
procedure normalize_symptom_weights (class : classname_type);
```
sets the sum of the weights of class to 1.0

```
function in_class_vector
   (class_name : classname_type; class_vct : class_vector) : boolean;
```
return true, iff class_name is in class_vct

```
procedure protocol
   (to_file : boolean; changer : double; i_requs : longint);
```
outputs a protocol of the testing session to a file (true) or onto the printer

```
procedure file_protocol_end;
```
closes the protocol file

```
procedure header_output (to_file : boolean);
```
outputs the test - preferences onto the testfile (true) or onto printer

```
function my_sqrt (arg : longint) : longint;
```
returns the square - root, if possible

```
function maximum (a, b : double) : double;
```
returns the maximum of a and b

```
function minimum (a, b : double) : double;
```
returns the minimum of a and b

B.5 Unit Training

B.5.1 Übersicht

- Verwendete Units:
  - Crt_Refi
  - Init
  - Basics
- Exportierte Funktionen : 2
- Exportierte Prozeduren : 2

B.5.2 Implementierung

```
interface
```
selection

```
procedure determine_tresholds (mode : double);
```
sets the values of the class tresholds to the means of the case similarity

```
function calc_simple_similarity
   (comp_case, index_case : case_type; class_index : classname_type)
   : double;
```
main function to determine the similarity between two cases

```
var
     result,
     e, w, u, n,
     omega, alpha_e : double;
     symptom_index : byte;

begin
   result := 0.0;
   e := 0.0; w := 0.0; u := 0.0; n := 0.0;

   inc (longtime_sim_counter);
   for statistical reasons only
   for symptom_index := 1 to nr_of_symptoms do
     begin
       omega := case_attribute_similarity
            (comp_case [symptom_index], index_case [symptom_index]);
       if omega = unknown_value
         then u := u + relevance_matrix [class_index, symptom_index]
         else if omega = redundant_value then n := n + 1.0
         else if omega <> both_not_there
             then
               begin
                 if omega >= theta
                   then e := e + omega*relevance_matrix
```

```
                        [class_index, symptom_index]
                  else w := w + (1.0 - omega) * relevance_matrix
                        [class_index, symptom_index]
               end; { if omega <> both_not_there }
     end; { for symptom_index }

   alpha_e := alpha * e;
   result := alpha_e / (alpha_e + beta * w + gamma * u + eta * n);

   calc_simple_similarity := result
end;


function calc_detailed_similarity
(comp_case, index_case : case_type; class_index : classname_type;
  var e, w, u, n, e_sum, w_sum : double;
  var e_num, w_num, u_num, n_num : byte;
  var sum_remainder, dist_sum_e, dist_sum_w, dist_sum_u : double)
     : double;
investigates the similarity between comp_case and index_case and sets
all used variables to change later the relevance weights

procedure training_phase;
executes the training phase-termination according to max_iterations or stable weights
begin
   ClrScr;
   highvideo;
   center ('Training phase',1);
   lowvideo;


   iteration_counter := 1;
   while (training_phase_iteration <> 0)
                    and (iteration_counter < max_iterations) do
       inc (iteration_counter);


   if protocol_on_testfile then file_protocol_end;
   highvideo;
   if iteration_counter >= max_iterations
     then center ('no stable weights observed ...',23)
     else center ('training phase successful ....',23);
   lowvideo;
end;


function training_phase_iteration : longint;
iteration of case comparisons and weight adaption / returns nr of changes
var
case_index,
```

```
     case_comp : longint;
     similarity_value,
     similarity_difference,
     e, w, u, n,
     x, y, z,
     e_new, w_new, u_new,
     e_sum, w_sum,
     sigma_e, sigma_w,
     sum_remainder,
     dist_sum_e, dist_sum_u,
     dist_sum_w,
     delta_relation,
     delta_w,
     omega : double;
     e_num, w_num,
     u_num, n_num : byte;
     index_case,
     comp_case : case_list_type;
     class_index : classname_type;
     index_case_values,
     comp_case_values : case_type;
     rel_show,
     symptom_index : byte;
     impossible_requires,
     changes : longint;
begin
   changes := 0;
   impossible_requires := 0;

   for case_index := 1 to nr_of_relevant_cases do
     begin
       index_case := case_array [case_index];
       generate_case_values
         (index_case_values, index_case.case_number);


       for case_comp := 1 to nr_of_relevant_cases do
           begin
             comp_case := case_array [case_comp];
             generate_case_values
                 (comp_case_values, comp_case.case_number);
             if not (common_classes
                   (comp_case.classes, index_case.classes))
               then
               for class_index := first_class to max_class_name do
                 if in_class_vector (class_index, comp_case.classes)
                   then
```

```
begin
  e_new := 0; w_new := 0; u_new := 0;
  sigma_e := 0.0; sigma_w := 0.0;
  sum_remainder := 0.0;
  delta_relation := 0.0;

  similarity_value :=
    calc_detailed_similarity
        (comp_case_values, index_case_values,
          class_index, w, u, n, e_sum, w_sum
         e_num, w_num, u_num, n_num, sum_remainder,
         dist_sum_e, dist_sum_w, dist_sum_u);
  if similarity_value >
    (treshold_array[class_index]+arithmetic_minimum)
      then
      begin
        similarity_difference :=
            similarity_value
            - treshold_array [class_index];
        case relevance_change of
          half_change :
              delta_relation := (1.0 /
                (treshold_array [class_index] +
                simmilarity_difference / 2)) - 1.0;
          normal_change :
              delta_relation := (1.0 /
                treshold_array [class_index]) - 1.0;
                that is just the same as : (1.0-δ_j)/δ_j
          double_change :
              delta_relation := maximum ((1.0 /
                (treshold_array [class_index]
                - similarity_difference / 2)) - 1.0,
                treshold_array [class_index] / 2);
          else error (5);
          end; { case }
          see subsection (5.1.4.1) for details
          e - z ⇒ e_new
          x contains the amount of different symptom weights
          wich have to be enlarged
          w_sum + x ⇒ w_new
          y contains the amount of unknown symptom weights
          wich have to be enlarged
          u + y ⇒ u_new
          calculates Σ_E and Σ_W
        for symptom_index := 1 to nr_of_symptoms do
          begin
```

```
    omega := case_attribute_similarity
         (comp_case_values [symptom_index],
          index_case_values [symptom_index]);
    if omega >= 0 then
      begin
        if omega >= theta
          then sigma_e := sigma_e +
            f_e (symptom_index, e_num, e,
            omega, relevance_matrix
              [class_index, symptom_index],
              dist_sum_e) / omega
        else sigma_w := sigma_w +
          f_w (symptom_index, w_num, w, omega,
          relevance_matrix [class_index,
           symptom_index], dist_sum_w)
           / (1.0-omega);
      end
  end;

e_new := relevance_change_formular
     (similarity_difference,similarity_value,
      e, w, u, n, w_num, u_num
      delta_relation, sum_remainder,
      sigma_w, sigma_e);
if e_new < 0 { impossible requirement }
then
  begin
    inc (impossible_requires);
    inc (hole_ir_counter);
  end
else
begin
  inc (hole_changes_counter);
  if u_num = 0
  then
    begin
      w_new := delta_relation *
         (alpha / beta) * e_new -
         (eta / beta) * n;
    end
  else
    begin
      if w_num = 0
        then
          begin
            u_new := delta_relation *
              (alpha / gamma) * e_new -
```

```
            (eta / gamma) * n;
          end
        else
          begin
            delta_w := (delta_relation *
              alpha * e_new - (gamma * u
               + eta * n + beta * w)) /
              (beta + gamma * distributor_c);
            w_new := w + delta_w;
            u_new := u +
              (distributor_c * delta_w);
          end
    end;

  now change weights in the relevance matrix
  inc (changes);
  for symptom_index := 1 to nr_of_symptoms do
  begin
    omega := case_attribute_similarity
       (comp_case_values [symptom_index],
        index_case_values [symptom_index]);
    if omega = unknown_value
      then
        relevance_matrix [class_index,
             symptom_index] :=
        f_u (symptom_index, u_num, u,
        relevance_matrix
         [class_index, symptom_index],
        dist_sum_u) * u_new
      else if (omega<>redundant_value) and
        (omega <> both_not_there) then begin
        if omega >= theta then
          relevance_matrix [class_index,
              symptom_index] :=
            f_e (symptom_index, e_num, e,
            omega, relevance_matrix
              [class_index, symptom_index],
            dist_sum_e)*e_new / omega
        else
            relevance_matrix [class_index,
            symptom_index] :=
            f_w (symptom_index, w_num, w,
            omega, relevance_matrix
            [class_index, symptom_index],
            dist_sum_w) * w_new / (1.0-omega);
    end; { else if .. }
```

```
                              end; { for symptom_index }
                            end; { e_new <> 0 }
                          end; { if similartiy_value > treshold_array }
                    end; { if in class vector }
            end; { for case_comp }
   training_phase_iteration := changes
end;
```

B.6 Unit Testing

B.6.1 Übersicht

- Verwendete Units:
  - Crt_Refi
  - Init
  - Basics
  - Training
  - Automati
- Exportierte Funktionen : 0
- Exportierte Prozeduren : 1

B.6.2 Implementierung

`interface`
selection

`procedure testing_phase (testfile_counter : longint);`
executes the testing phase according to `(5.1.5)`

```
var
     good, bad,
     casebase_index,
     inner_casebase_index : word;
     goodies, badies : class_data;
     test_case_values,
     case_in_base_values : case_type;
     class_belonging,
     classified_vector : class_vector;
     case_in_base : case_list_type;
     symptom_index,
     line_nr : byte;
     class_index,
     classified : classname_type;
     max_simil, temp_simil,
     result : double;
     case_count,
     nr_of_testcases : longint;
begin
   ClrScr;
   highvideo;
   if probability_testcase = 1.0
     then center ('Testing phase with loss of information
          - boundary cases only',1)
```

```
  else center ('Testing phase with loss of information',1);
lowvideo;

nr_of_testcases := nr_of_all_cases * testcase_factor;
good := 0; bad := 0; casebase_index := 0;
for class_index := first_class to max_class_name do
  begin
    goodies [class_index] := 0;
    badies [class_index] := 0;
  end;

center ('test classification : ',5);
gotoxy (26, 7);
write ('training set size ',
     nr_of_relevant_cases / bco_base_size*100:6:2,' %');
gotoxy (26, 9);
write ('noise probability ',noise_probability*100:6:2,' %');
gotoxy (25, 20);
write ('still ',100:3,' % cases to test ...');
gotoxy (24, 12); write ('correct : ',good:5);
gotoxy (41, 12); write ('wrong : ',bad:5);

for case_count := 1 to nr_of_testcases do
  begin
    inc (casebase_index);
    if casebase_index > nr_of_all_cases then casebase_index := 1;
    generate_case_values (test_case_values, casebase_index);
    class_belonging := determine_case_classes (test_case_values);
    if class_belonging <> classless then
      begin
        for symptom_index := 1 to nr_of_symptoms do
        if random < noise_probability then
             test_case_values [symptom_index] := unknown;
        destroys a certain part of case information ...
        max_simil := -0.1;
        for inner_casebase_index := 1 to nr_of_relevant_cases do
            begin
              case_in_base := case_array [inner_casebase_index];
              generate_case_values
                   (case_in_base_values, case_in_base.case_number);
              for class_index := first_class to max_class_name do
              if in_class_vector
                   (class_index, case_in_base.classes) then
                begin
                  temp_simil := calc_simple_similarity
                       (case_in_base_values, test_case_values,
                        class_index);
                  if max_simil < temp_simil
```

```
                    then
                      begin
                        max_simil := temp_simil;
                        classified := class_index;
                        classified_vector := case_in_base.classes;
                      end; { if max simil < temp simil }
                  end; { if in class vector }
              end; { for inner case base index }
          if in_class_vector (classified, class_belonging)
          then
            begin
              if max_simil > 0 then
                begin
                  inc (good);
                  gotoxy (34, 12);
                  write (good:5);
                  inc (goodies [classified]);
                end;
            end
          else
            begin
              inc (bad);
              gotoxy (51, 12);
              write (bad:5);
              inc (badies [classified]);
            end;
        end; { if not classless }
      gotoxy (31, 20);
      write ((1.0-case_count / nr_of_testcases)*100:3:0);
    end; { for case count }
  if (good+bad) > 0
    then result := good/(good+bad)*100
    else result := 0;
  gotoxy (31, 15);
  write ('result : ');
  highvideo;
  write (result:6:2);
  lowvideo;
  writeln (' %');
end;
```

B.7 Unit Automatic

B.7.1 Übersicht

- Verwendete Units:
  - Crt_Refi
  - Init
  - Basics
  - Training
- Exportierte Funktionen : 0
- Exportierte Prozeduren : 10

B.7.2 Implementierung

```
interface
```
selection

```
procedure init_casebase_dimensions;
```
initializes nr_of_symptoms and nr_of_attrib_values randomly

```
procedure define_random_casebase;
```
generates a random casebase

```
procedure set_global_variables (global_variable_setting : byte);
```
sets alpha, beta, gamma and eta

```
begin
   alpha := 1.0;
   case global_variable_setting of
         0 : begin beta := 0.1; gamma := 0.1; eta := 0.1; end;
         1 : begin beta := 10; gamma := 10; eta := 10; end;
         2 : begin beta := 0.1; gamma := 1; eta := 1; end;
         3 : begin beta := 10; gamma := 1; eta := 1; end;
         4 : begin beta := 1; gamma := 10; eta := 1; end;
         5 : begin beta := 1; gamma := 0.1; eta := 1; end;
         6 : begin beta := 1; gamma := 1; eta := 10; end;
         7 : begin beta := 1; gamma := 1; eta := 0.1; end;
         8 : begin beta := 1; gamma := 1; eta := 1; end;
         else error (2);
         end; { case }
end;
```

```
procedure set_casebase_filling (casebase_filling : byte);
```
calculates the percentage for the testcase probability

```
procedure get_means_of_tresholds (tresh_calc : treshold_calc_types);
```
claculates deviation, means and number of class elements

```
procedure set_noise (noise : byte);
```
*sets the global variables for test with noise * 10 %*

```
procedure store_presettings (test_nb : longint);
```
opens a file and stores all important casebase dates

```
procedure store_results
(test_nb : longint; result : double; goodies, badies : class_data);
```
appends to the protocol file the results of a test

```
procedure store_training_results (test_nb : longint);
```
stores all important training results

```
procedure shrink (test_nb : longint);
```
calls another pgm to compress the data in the files

Anhang C

Literaturverzeichnis

Literaturverzeichnis

[Aha91] David W. Aha. Case-Based Learning Algorithms. in: Proc. DARPA Workshop on Case Based Reasoning, Morgan Kaufmann. Pub. (1991), 147-157

[AKA91] David W. Aha, Dennis Kibler, Marc K. Albert. Instance-Based Learning Algorithms. Machine Learning 6 (1991), 37-66

[And89] Anderson, John Robert. Kognitive Psychologie. Spektrum der Wissenschaft Verlagsgesellschaft (1989)

[AS83] Dana Angluin, Carl H. Smith. A Survey of Inductive Inference: Theory and Methods. Computing Surveys 15 (1983), 237-269

[ASS87] Harold Abelson, Gerald Jay Sussman, Julie Sussman. Structure and Interpretation of Computer Programs. Fifth reprinting. MIT Press (1987)

[AWB92] Klaus-Dieter Althoff, Stefan Weß, Brigitte Bartsch-Spörl, Dietmar Janetzko, Frank Maurer, Angi Voß. Fallbasiertes Schließen in Expertensystemen: Welche Rolle spielen Fälle für wissensbasierte Systeme? in: KI-Künstliche Intelligenz, 4, FBO-Verlag (1992)

[AW91a] Klaus-Dieter Althoff, Stefan Weß. Case-Based Knowledge Acquisition, Learning and Problem Solving for Diagnostic Real World Tasks. in: Proc. European Knowledge Acquisition Workshop, (EKAW-91), Crieff, Scottland (1991)

[AW91b] Klaus-Dieter Althoff, Stefan Weß. Fallbasiertes Problemlösen in Expertensystemen - begriffliche und inhaltliche Betrachtungen, SEKI Working Paper SWP-91-03, Universität Kaiserslautern (1991)

[AW92a] Klaus-Dieter Althoff, Stefan Weß. Ähnlichkeit in PATDEX. in Workshop: Ähnlichkeit von Fällen beim fallbasierten Schließen, SEKI Working Paper SWP-92-11, Universität Kaiserslautern (1992)

[AW92b] Klaus-Dieter Althoff, Stefan Weß. Case-Based Reasoning and Expert System Development. in: Schmalhofer, Strube, Wetter (Hrsg.), Contemporary Knowledge Engineering and Cognition, Springer Verlag (1992)

[Boe92] Katy Börner. Positionspapier. Ein allgemeines Modell zur Speicherung episodischen Wissens und separat lernbare Ähnlichkeitsmaße (1992)

[BR84] Banahan, A. Rutter. UNIX lernen, verstehen, anwenden. Carl Hanser Verlag München Wien (1984)

[CF82] Paul R. Cohen, Edward A. Feigenbaum. The Handbook of Artificial Intelligence. Volume 3. Pitman (1982)

[FT71] W. W. Fairchild, C. Ionescu Tulcea. Topology. W. B. Sounders Company USA (1971)

[GK79] Leo A. Goodman, William H. Kruskal. Measures of Association for Cross Classifications. Springer-Verlag New York Heidelberg Berlin (1979)

[Gol84] Adele Goldberg. SMALLTALK-80, The Interactive Programming Environment. Addison-Wesley (1984)

[GR85] Adele Goldberg, David Robson. SMALLTALK-80, The Language and its Implementation. Reprinting. Addison-Wesley (1985)

[Jan91a] Klaus P. Jantke. Case Based Learning and Inductive Inference. GOSLER. BMFT-Verbundprojekt (1991)

[Jan91b] Klaus P. Jantke. Monotonic and Non-monotonic Inductive Inference. New Generation Computing 8 (1991) 4, 349-360

[JB81] Klaus P. Jantke, Hans-Rainer Beick. Combining Postulates of Naturalness in Inductive Inference. ELK 17 (1981) 8/9, 465-484

[JWM93] Dietmar Janetzko, Stefan Weß & E. Melis. Goal Driven Similarity Assessment. in: Hans-Jürgen Ohlbach (Hrsg.), Proc. German Workshop on AI (GWAI-1992), Bonn, Springer Verlag (1993)

[KM79] Albert K. Kurtz, Samuel T. Mayo. Statistical Methods in Education and Psychology. Springer Verlag (1979)

[Kol83] Janet L. Kolodner. Reconstructive Memory: A Computer Model. Cognitive Science, 7:(1983), 281-328

[Kop91] Helmut Kopka. LaTex - Eine Einführung. Addison-Wesley (1991)

[Mic83] Michalski, R.S. et al. Machine Learning: An Artificial Intelligence Approach. Tiga Publishing (1983)

[Mol92] Michael Molter. ZEBRA - Ein System zur Zeichenklassifikation für eingeschränkte Fontfamilien. Diplomarbeit, DFKI Kaiserslautern (1992)

[Ric88] Elain Rich, KI-Einführung und Anwendungen. MC Graw Hill (1988)

[Ric89] Michael M. Richter. Prinzipien der Künstlichen Intelligenz. B. G. Teubner Stuttgart (1989)

[Ric90] Michael M. Richter. Konnektionismus. Vorlesungsskriptum Universität Kaiserslautern (1990)

[Ric91] Michael M. Richter. Lernende Systeme. Vorlesungsskriptum Universität Kaiserslautern (1991)

[Ric92] Michael M. Richter. Classification and Learning of Similarity Measures. in Proc. der Jahrestagung der Gesellschaft für Klassifikation. (ed). Opitz, Lassen, Klar (Hrsg.), Studies in Classification, Data Analysis and Knowledge Organisation, Springer Verlag (1992)

[RW91] Michael M. Richter, Stefan Weß. Similarity, Uncertainty and Case-Based Reasoning in PATDEX. in: Boyer, R.S. (Ed.), Automated Reasoning, Essays in Honor of Woody Bledsoe, Kluwer Academic Publishers (1991)

[Sal88] Steven Salzberg. Exemplar-Based Learning: Theory and Implementation (Technical Report TR 10-88). Cambridge, MA: Harvard University, Center for Research in Computing Technology (1988)

[Sch82] Roger C. Schank. Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press, New York (1982)

[Sch77] Jürgen Schürmann. Polynomklassifikatoren für die Zeichenerkennung. Oldenburg Verlag (1977)

[Sta91] Michael Stadler. Vergleich von fallbasierten, induktiven und statistischen Lernverfahren für die Klassifikation. Diplomarbeit. Universität Kaiserslautern (1991)

[SW86] Craig Stanfill und David Waltz. Toward Memory-Based Reasoning. Communication of the ACM, 29 (12): (1986), 1213-1229

[TP91] Turbo Pascal 6.0 Programmier-Handbücher (Band 1 - 4). Borland (1991)

[Tve77] Amos Tverski. Features of Similarity. Psychological Review (1977), 327-352

[Wes91] Stefan Weß. PATDEX/2 - ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen. SEKI Working Paper SWP91/01, Universität Kaiserslautern (1991)

[Wes93] Stefan Weß. PATDEX - Ein Ansatz zur wissensbasierten und inkrementellen Verbesserung von Ähnlichkeitsurteilen in der fallbasierten Diagnostic. in: Proc. der 2. Deutschen Tagung Expertensysteme, XPS-93, Springer Verlag (1993)

[WPA92] Stefan Weß, J. Paulokat, K.-D. Althoff. Fallbasiertes Schließen - ein Überblick. Technical Report, Fachbereich Informatik, Universität Kaiserslauten (1992)