Fallbasiertes Schließen zur Kreditwürdigkeitsprüfung

Wolfgang Wilke, Ralph Bergmann, Klaus-Dieter Althoff

In diesem Artikel diskutieren wir Anforderungen aus der Kreditwürdigkeitsprüfung und ihre Erfüllung mit Hilfe der Technik des fallbasierten Schließens. Innerhalb eines allgemeinen Ansatzes zur fallbasierten Systementwicklung wird ein Lernverfahren zur Optimierung von Entscheidungskosten ausführlich beschrieben. Dieses Verfahren wird, auf der Basis realer Kundendaten, mit dem fallbasierten Entwicklungswerkzeug INRECA empirisch bewertet. Die Voraussetzungen für den Einsatz fallbasierter Systeme zur Kreditwürdigkeitsprüfung werden abschließend dargestellt und ihre Nützlichkeit diskutiert.

1. Einleitung und Motivation

Dieser Beitrag beschäftigt sich mit dem finanzwirtschaftlich interessanten Bereich der Kreditwürdigkeitsprüfung mit Hilfe der vergleichsweise neuen Technologie des fallbasierten Schließens (Engl.: Case-Based Reasoning; abgekürzt: CBR; z.B. Bartsch-Spörl & Wess, 1996). Hierbei handelt es sich um einen Ansatz zum Lösen neuer Probleme durch Anpassen der Lösungen früherer, ähnlicher Probleme.

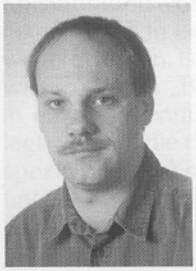
Aus dieser Sicht ist die Kreditwürdigkeitsprüfung eine typische Klassifikationsaufgabe. Aufgrund verschiedener Merkmale eines potentiellen Kreditnehmers, wie beispielsweise seines Zahlungsverhaltens, erfolgt eine Bewertung des Kunden hinsichtlich seiner Kreditwürdigkeit. Der Kunde muß demzufolge einer der beiden Klassen *kreditwürdig* oder *nicht kreditwürdig* zugeordnet werden. Wir betrachten hier die Situation, daß der potentielle Kunde ein Wirtschaftsunternehmen ist. Die beiden möglichen Klassen heißen in diesem Bereich *solvent* oder *insolvent*. Dabei können zwei prinzipielle Ansätze unterschieden werden (Günther & Scheipers, 1993).

Im Rahmen des *fundamentalen Ansatzes* wird versucht, über die Abschätzung aller relevanten außerbetrieblichen Einflußfaktoren (z.B. Branchensituation, Konkurrenzsituation, Marktentwicklung) und innerbetrieblichen Einflußfaktoren (z.B. Produktstruktur, Ertragssituation, Management) ein Bild von der Unternehmensentwicklung zu erhalten. Aufgrund dieses resultierenden Unternehmensprofils wird anschließend die Kreditwürdigkeit bewertet.

Dagegen geht der *technische Ansatz* von einer Jahresabschlußanalyse aus und gelangt über Kennzahlenvergleiche zu Aussagen über die Insolvenzwahrscheinlichkeit und damit zu einer Bewertung der Kreditwürdigkeit. Die technischen Ansätze basieren in der Regel auf Methoden, die mit Hilfe einer Trainingsmenge bereits bekannter Fälle diejenigen Kombinationen von Kennzahlen finden, die solvente und insolvente Unternehmen möglichst scharf bei bestimmten Werten trennen.

Der Großteil bislang vorliegender empirischer Studien ist den technischen Ansätzen zuzuordnen, da die fundamentalen Ansätze aufgrund der Vielzahl der zu verarbeitenden Informationen und der Subjektivität der Wertung als Basis empirischer Untersuchungen weitaus schwieriger zu handhaben sind (Günther & Scheipers, 1993). Innerhalb der technischen Insolvenzprognose können verschiedene Verfahren aus der Statistik und der Mustererkennung (z.B. Regressionsanalyse, Diskriminanzanalyse, Clusteringtechniken: Heno, 1983; Rösler, 1988) bzw. der Künstlichen Intelligenz eingesetzt werden. Einen Schwerpunkt bei letzterem bilden vor allem die künstlichen neuronalen Netze (Baetke & Krause, 1994; Berndt, 1995; Beuter, Reiss & Rust, 1994); es gibt allerdings auch eine Reihe von Anwendungen, die dem Bereich der wissensbasierten Systeme zuzuordnen sind (Mertens, Borkowski & Geis, 1993; Schwarze & Rosenhagen, 1993; Althoff, Auriol et al., 1995).

Die Grundlage der vorliegenden Arbeit ist eine Studie mit einer Schweizer Großbank, die zum Gegenstand hatte, die Verwendbarkeit des fallbasierten Schließens am Beispiel des INRECA[1]-Systems für die Kreditwürdigkeitsprüfung von Wirtschaftsunternehmen zu testen (Reinartz & Wilke, 1995). Aufbauend auf den Erkenntnissen aus dieser Studie wurde das in



Wolfgang Wilke studierte an der Universität Kaiserslautern Informatik und ist dort seit Mai 1995 als wissenschaftlicher Mitarbeiter in den Projekten INRECA und WiMo tätig. Zur Zeit beschäftigt er sich im ESPRIT-Projekt INRECA-II mit der Wissensmodellierung und Adaption in fallbasierten Systemen.

Dr. Ralph Bergmann promovierte an der Universität Kaiserslautern im Bereich des fallbasierten Problemlösens. Seine Arbeitsgebiete umfassen fallbasiertes Schließen, maschinelles Lernen und Wissensakquisition für Entscheidungsunterstützungs- und Planungsaufgaben. Zur Zeit ist er Projektleiter im ESPRIT-Projekt INRECA-II sowie im Projekt WiMo für die Universität Kaiserslautern.

Dr. Klaus Dieter Althoff promovierte über fallbasiertes Lernen im Rahmen des MOLTKE-Systems zur Diagnose technischer Systeme. Er war Projektleiter im Esprit-Projekt INRECA („Induction and Reasoning from Cases") für die Universität Kaiserslautern. Sein derzeitiger Forschungsschwerpunkt ist die Beschreibung, Analyse und Entwicklung fallbasierter Systeme.

INRECA verwendete Klassifikationsverfahren erweitert, so daß nicht die *Optimierung der Korrektheit der Klassifikation gegebener Falldaten* (Optimierung der Entscheidungskorrektheit) primäres Ziel ist, sondern die *Minimierung der durch Fehlklassifikation entstehenden Kosten* (Optimierung der Entscheidungskosten).

2. Grundlagen des fallbasierten Schließens

Fallbasiertes Schließen ist ein Ansatz zum Lösen neuer Probleme durch Anpassen der Lösungen früherer, ähnlicher Probleme (Aamodt & Plaza, 1994; Althoff, Auriol et al., 1995). Ein *Fall* besteht dabei zumindest aus einer Problem- sowie einer Lösungsbeschreibung. Im Rahmen der Kreditwürdigkeitsprüfung relevante Problemcharakteristika sind z.B. absolute Werte aus der Bilanz oder dem Jahresabschluß, Kennzahlen über diesen Werten, bewertete Kennzahlen bzw. externe Faktoren. Mögliche Lösungen, falls als potentielle Kunden Wirtschaftsunternehmen betrachtet werden, sind die Klasseneinteilungen *solvent* bzw. *insolvent*.

Fälle werden nun innerhalb einer *Fallbasis* gespeichert, die man sich auch vereinfacht als eine Datenbank von Fällen vorstellen kann. Darüber hinaus wird ein *Ähnlichkeitsbegriff* zwischen Problemstellungen benötigt. Probleme werden dabei sozusagen als „Schlüssel in die Datenbank" verwendet. Zentral ist hierbei, neben dem für Datenbanken üblichen exakten Matching, die Verwendung ähnlichkeitsbasierten Matchings auf Basis des oben erwähnten Ähnlichkeitsbegriffes.

Ist nun ein neues Problem zu lösen und ist keine analytische (direkte) Lösung verfügbar (z.B. weil diese nicht zu realisieren oder zu teuer ist), dann beinhaltet die Vorgehensweise des fallbasierten Schließens, innerhalb der Fallbasis ein ähnliches (bereits gelöstes) Problem zu suchen und dessen Lösung als Ausgangspunkt zur Lösung des neuen Problems zu nutzen.

Wichtige Fragestellungen im fallbasierten Schließen sind (Wess, 1995):

- Welche Fälle kommen in die Fallbasis, d.h. welche Aufnahmestrategie wird gewählt?
- Welche Attribute repräsentieren einen Fall und mit welchem Attributgewicht gehen sie in die Ähnlichkeitsberechnung ein?
- Was ist Ähnlichkeit?
- Wie sucht man effizient nach ähnlichen Fällen?
- Was ist eine Lösung/nützliche Information? (Althoff, Richter & Wilke, 1996).
- Wie adaptiert man eine Lösung möglichst nutzbringend? (Voss, Bartsch-Spörl & Oxman, 1996)

Fallbasiertes Schließen kann nun, sozusagen als Spezialfall, auf die Klassifikation angewendet werden. Mit Hilfe eines Ähnlichkeitsmaßes wird nach Maßgabe der „Regel des nächsten Nachbarn" ein ähnlicher Fall ausgewählt und dessen Klasse als Lösung übernommen.

Dies bedeutet, daß fallbasiertes Schließen mit anderen Klassifikationsmethoden in Konkurrenz steht: Entscheidungsbäume, neuronale Netze, statistische Verfahren (vgl. Kapitel 6). Darüber hinaus ist es aber mit fallbasiertem Schließen auch möglich, allgemeinere Entscheidungsunterstützungsaufgaben zu behandeln (Althoff & Bartsch-Spörl, 1996; Ehrenberg, 1996), wo z.B. die Problembeschreibung unvollständig und die intendierte Art der Lösung nur grob vorgegeben ist. Ebenso sind bereits eine Reihe von fallbasierten Ansätzen zur Behandlung von Entwurfs- (vgl. Oxman & Voss, 1996) und Planungsaufgaben (vgl. Bergmann, Munoz & Veloso, 1996) entwickelt worden.

3. Entwicklung fallbasierter Systeme

Um den Aufwand zur Entwicklung eines fallbasierten Systems in vertretbaren Grenzen zu halten, hat es sich als vorteilhaft erwiesen, eine verfügbare CBR-Shell einzusetzen[2]. Hierdurch kann sich die Entwicklung auf die Daten- und Wissensmodellierung für die entsprechende Anwendung und die Integration in die im Unternehmen vorhandene DV-Umgebung beschränken. Nachfolgend betrachten wir Vorgehensweisen, die eine systematische Modellierung ermöglichen sollen.

3.1 Designentscheidungen bei der Entwicklung fallbasierter Klassifikationssysteme

Typischerweise erfordert die Entwicklung fallbasierter Klassifikationssysteme (wie z.B. zur Kreditwürdigkeitsprüfung) einen geringeren Entwicklungsaufwand, als die Entwicklung fallbasierter Design- oder Planungsanwendungen. Doch selbst im Rahmen der Entwicklung fallbasierter Klassifikationssysteme sind eine Reihe von Designentscheidungen zu treffen, wie z.B. die Auswahl von Attributen und Wertebereichen zur Fallrepräsentation, die Bestimmung von Ähnlichkeitsmaßen, die Festlegung von Attributgewichten und die Auswahl der zu speichernden Fälle. Richter (1995) spricht in diesem Zusammenhang von vier verschiedenen „Wissenscontainern" (Vokabular, Ähnlichkeitsmaß, Fallbasis, Anpassungswissen), die verschiedene Arten von Wissen enthalten. Der Inhalt dieser Container determiniert das endgültige Systemverhalten. Das Ziel einer Entwicklungsmethodologie für fallbasierte Systeme besteht nun darin, ingenieursmäßige Vorgehensweisen zur Festlegung dieser Containerinhalte zu finden. Diese müssen sowohl zu einem Systemverhalten führen, das den vielfältigen Anforderungen, die in einer speziellen Anwendung zu erfüllen sind, gerecht wird, als auch mit einem möglichst geringen Aufwand bestimmt werden können, um die Kosten für die Systementwicklung niedrig zu halten (vgl. auch Althoff & Aamodt, 1996, bzw. Althoff, 1996, hinsichtlich der Behandlung von Domänenanforderungen und ihre Auswirkungen auf die Architektur fallbasierter Systeme).

3.2 Inkrementelle Systementwicklung

In jüngster Zeit hat sich eine inkrementelle Entwicklungsstrategie als vorteilhaft herausgestellt, bei der das endgültige System in einer Folge von Prototypen entwickelt wird (Bartsch-Spörl, 1996). Ein solcher Prototyp umfaßt hierbei eine spezielle Wahl der Containerinhalte; für Klassifikationssysteme also Attribute, Ähnlichkeitsmaße, Gewichte und Fallbasis. Entwickelte Prototypen werden dabei in bezug auf die Anforderungen aus der Applikation bewertet. Aufgrund dieser Bewertung

[1] „Induction and Reasoning from Cases" (Esprit-Projekt, gefördert unter Nr. 6322 von Mai 1992 bis Oktober 1995). Partner sind AcknoSoft (Paris; Koordinator), Irish Medical Systems (Dublin), tecInno (Kaiserslautern) und die Universität Kaiserslautern.

[2] Ein Überblick über die auf dem Markt verfügbaren CBR-Shells findet sich in Althoff, Auriol et al. (1995).

wird dann entschieden, ob bereits ein geeignetes Zielsystem vorliegt oder ob ein verbesserter Prototyp entwickelt werden muß und welcher Weg hierzu eingeschlagen werden sollte.

Die Bewertung muß verschiedene Aspekte berücksichtigen, wie z.B. Entscheidungskorrektheit, Entscheidungsqualität, Entscheidungskosten, oder auch das Antwortzeitverhalten. In der Kreditwürdigkeitsprüfung spielen bei der Entwicklung eines Systems die Kosten, die durch eine korrekte bzw. falsche Klassifikation (Einordnung der Kreditnehmer in solvente und insolvente Kunden) verursacht werden, eine übergeordnete Rolle. Kosten sind hierbei solche, die dem Kreditinstitut aufgrund einer falschen Entscheidung entstehen (Verlust des Kreditvolumens bzw. Verzicht auf den Zinsertrag). In diesem Fall besteht folglich das Ziel der Entwicklung darin, ein System zu erhalten, das möglichst geringe Entscheidungskosten verursacht.

3.3 Darstellung des Entwicklungsprozesses in einem Graphen

Den Prozeß der Entwicklung eines fallbasierten Systems kann man durch einen gerichteten Graphen verdeutlichen, den wir Entwicklungsgraph nennen (siehe Abbildung 1). Ein Knoten eines Entwicklungsgraphen beschreibt einen möglichen Prototyp, und eine gerichtete Kante spezifiziert einen möglichen Entwicklungsschritt auf dem Weg zum endgültigen System. Ein Entwicklungsschritt kann hierbei zum Beispiel die Aufnahme eines weiteren Attributes in die Fallbeschreibung bezeichnen oder die Anpassung des Attributgewichts. Im allgemeinen sind eine Vielzahl unterschiedlicher Entwicklungs- oder Anpassungsschritte erforderlich.

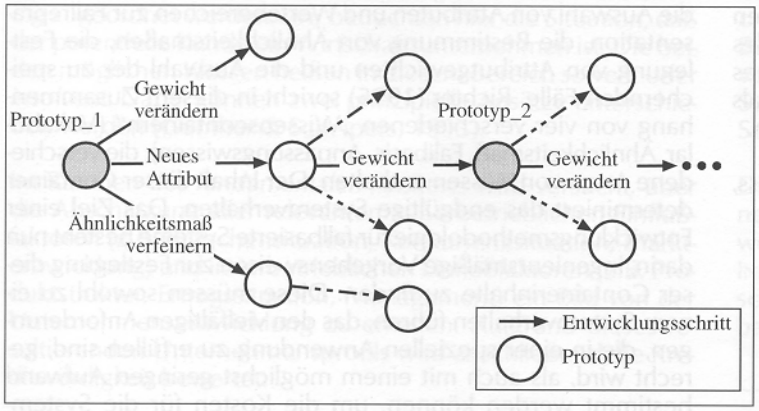


Abbildung 1: Beispiel eines Entwicklungsgraphen

In diesem Modell bedeutet Systementwicklung Wegesuche bzw. kontrollierte Navigation im Entwicklungsgraphen. Hierbei ist man natürlich nicht nur daran interessiert, überhaupt zu einem System zu gelangen, das den gestellten Anforderungen gerecht wird, sondern man ist insbesondere an einem Weg interessiert, der geringe Entwicklungskosten verursacht. Hierbei ist zu beachten, daß unterschiedliche Entwicklungsschritte mit unterschiedlichen Kosten verbunden sind. Die Kosten zur Erweiterung der Fallbeschreibung um ein weiteres Attribut (z.B. "Betriebsgröße" als Entscheidungskriterium bei der Kreditwürdigkeitsprüfung) können beträchtlich sein, wenn die erforderliche neue Information zuvor für alle in der Fallbasis befindlichen Fälle erhoben und dem System verfügbar gemacht werden muß. Hingegen ist die Veränderung eines Attributgewichts in der Regel mit einem geringeren Entwicklungsaufwand verbunden, da lediglich die Berechnungsfunktion für die Ähnlichkeit (z.B. Parameteränderung) anzupassen ist.

3.4 Methoden zur Unterstützung der Systementwicklung

Ein weiterer wesentlicher Faktor, der die Entwicklungskosten eines fallbasierten Systems beeinflußt, betrifft den Automatisierungsgrad der Navigation durch den Entwicklungsgraphen. Je besser die Entscheidungsfindung für den nächsten Entwicklungsschritt durch Entwicklungswerkzeuge unterstützt werden kann, um so weniger Fehlentscheidungen im Entwicklungsgraphen - die ja immer mit zusätzlichen Kosten verbunden sind - müssen in Kauf genommen werden. Die beste Unterstützung, die ein solches Entwicklungswerkzeug bieten kann, besteht in der vollautomatischen Navigation im Entwicklungsgraphen. Diese ist dann möglich, wenn zumindest einzelne Entwicklungsschritte automatisch durchgeführt werden können und wenn operationale Kriterien für die Auswahl eines Entwicklungsschrittes formulierbar sind. In diesem Fall kann man heuristische Suchverfahren im Entwicklungsgraphen zur Unterstützung der Systementwicklung einsetzen, um den Aufwand der Entwicklung zu verringern.

4. Fallstudie: Optimierung von Entscheidungskosten in der Kreditwürdigkeitsprüfung

Nachfolgend stellen wir einen Ansatz vor, der durch heuristische Suche im Entwicklungsgraphen die Systementwicklung für eine Anwendung in der Kreditwürdigkeitsprüfung unterstützt. Dazu wollen wir zuerst die Funktionsweise der fallbasierten Klassifikation näher betrachten und anschließend unsere Repräsentation von Entscheidungskorrektheit und Entscheidungskosten vorstellen.

Wir beschreiben dann einen Algorithmus, der durch Lernen von Attributgewichten die Entscheidungskosten, die ein fallbasiertes Klassifikationssystem verursacht, minimiert.

4.1 Fallbasierte Klassifikation mit den k-nächsten Nachbarn

In einem fallbasiertem System für analytische Aufgaben besteht ein Fall $c=(f_1,...f_n,t_i)$ aus n beschreibenden Attributen f und aus der Klasse des Falles t_i. Die Menge $T=\{t_1,...,t_m\}$ stellt alle möglichen Lösungsklassen in der jeweiligen Anwendung dar. Die Fallbasis FB ist definiert als die Menge aller Fälle, die aus der Vergangenheit bekannt sind. Wird nun eine neue Anfrage $q=(q_1,...q_n)$ an die Fallbasis gestellt, werden die k ähnlichsten Fälle zum Anfragefall q aus der Fallbasis vom System zurückgeliefert. Die Ähnlichkeit sim(q,c) zwischen einer Anfrage q und einem Fall c aus der Fallbasis FB berechnet sich aus:

$$sim(q,c) = \sum_{a=1}^{n} w_a * sim_a(q_a, f_a)$$

wobei w_a das Gewicht für Attribut f_a und $sim_a(q_a,f_a)$ das „lokale Ähnlichkeitsmaß" für das Attribut f_a darstellt. Das fallbasierte System klassifiziert die Klasse der Anfrage, indem es die k nächsten Nachbarn $K=\{r_1,...,r_k\}$ zur Anfrage berechnet und die am häufigsten vorkommende Klasse in diesen Fällen als Klasse des Anfragefalls wählt.

Wir wollen nun betrachten, wie sich Entscheidungskorrektheit und Entscheidungskosten für ein fallbasiertes System repräsentieren lassen.

4.2 Repräsentation von Entscheidungskorrektheit und Entscheidungskosten

Die Entscheidungskorrektheit ist eine Anforderung, die sich auf das Verhältnis der richtigen und falschen Vorhersagen bei der Klassifikation bezieht. In der Kreditwürdigkeitsprüfung gibt es zwei verschiedene Klassen, die für solvente bzw. insolvente Kunden stehen. Eine korrekte Entscheidung des Systems liegt vor, falls ein solventer Bankkunde als solvent oder ein insolventer Kunde als insolvent vom Klassifikationssystem eingeschätzt wird. Bei der Einschätzung eines insolventen Kunden als solvent, bzw. eines solventen Kunden als insolvent, handelt es sich folglich um eine falsche Klassifikation. Die Entscheidungskorrektheit bezeichnet nun das Verhältnis der Anzahl der richtigen Entscheidungen zu der Anzahl der insgesamt vom System getroffenen.

Eine übersichtliche Möglichkeit die Entscheidungskorrektheit von Klassifikationssystemen darzustellen, ist eine Confusionmatrix (Weiss & Kulikowski, 1991, S.18), wie in der Tabelle 1 dargestellt.

$P_{i,j}$	vorhergesagte Klasse	
Klasse des Anfragefalls	**solvent**	**insolvent**
solvent	0,275	0,225
insolvent	0,125	0,375

Tabelle 1: Eine Confusionmatrix für die verschiedenen Klassifikationswahrscheinlichkeiten

Jeder Eintrag $p_{i,j}$ der Matrix bezeichnet die Wahrscheinlichkeit, daß für eine Klasse eines Anfragefalls eine Klasse vom Klassifikationssystem vorhergesagt wird. In unserem Beispiel wird für einen Anfragefall der Klasse „solvent" in 27,5 Prozent der Fälle auch die Klasse „solvent" vom System vorhergesagt und in 22,5 Prozent der Fälle die Klasse „insolvent". Die Wahrscheinlichkeiten für richtige Klassifikationen befinden sich in der Diagonalen der Matrix und die restlichen Einträge bezeichnen die Wahrscheinlichkeiten für die jeweiligen falschen Klassifikationen. Die Entscheidungskorrektheit eines Systems wird also gerade durch das Verhältnis der Summe der Werte in der Diagonalen der Matrix zu den übrigen Werten ausgedrückt.

Eine Möglichkeit zur Messung dieser Wahrscheinlichkeiten bei einem CBR-System für eine Menge von Fallbeispielen ist ein Leave-one-out-Test (Weiss & Kulikowski, 1991, S.31). Dabei wird jeder Fall aus der Trainingsmenge als eine Anfrage an das fallbasierte System gestellt, das alle Fälle bis auf den Anfragefall in der Fallbasis enthält. Daraus berechnen sich dann die Klassifikationswahrscheinlichkeiten für die jeweilige Trainingsmenge.

Nachdem wir gesehen haben, wie man Entscheidungskorrektheit repräsentieren kann, wollen wir eine Möglichkeit zur Repräsentation von Entscheidungskosten vorstellen.

Die Entscheidungskosten, die durch ein Klassifikationssystem verursacht werden, kann man in einer ähnlich aufgebauten Matrix, die wir Matrix der Entscheidungskosten nennen, darstellen. Die Matrix in der Tabelle 2 zeigt hier mögliche Entscheidungskosten für das Beispiel der Kreditwürdigkeitsprüfung. Diese Einträge können, je nach Einsatzszenario für das ein System optimiert werden soll, natürlich stark variieren.

$C_{i,j}$	vorhergesagte Klasse	
Klasse des Anfragefalls	**solvent**	**insolvent**
solvent	-1	1
insolvent	10	-10

Tabelle 2: Ein Beispiel für mögliche Entscheidungskosten einer Klassifikation in der Insolvenzprognose

Hier beschreiben die Einträge $C_{i,i}$, die in der Diagonalen stehen, den Gewinn von richtigen Entscheidungen und alle anderen Einträge die Kosten für falsche Klassifikationen. Wird vom Klassifikationssystem ein insolventer Kreditkunde als solvent eingestuft, verliert die Bank wahrscheinlich einen großen Teil der gesamten Kreditsumme. Stuft das System jedoch einen solventen Kunden als insolvent ein, so verliert die Bank nur die Zinseinkünfte, da das Kreditgeschäft nicht abgeschlossen wird. Die beiden Einträge $C_{2,1}$ und $C_{1,2}$ in der Tabelle, repräsentieren diese unterschiedlichen Kosten, die durch falsche Klassifikationen verursacht werden können. Der Nutzen durch eine richtige Entscheidung des Systems steht in den Diagonalfeldern der Matrix.

Eine ähnliche Darstellung findet sich bei (Weiss & Kulikowski, 1991, S. 21) und (Michie, Spiegelhalter & Taylor, 1994, S. 224), wo jedoch nur die Kosten von falschen Entscheidungen berücksichtigt werden.

Mit der Confusionmatrix und der Matrix der Entscheidungskosten können wir nun den Erwartungswert für die gesamten Entscheidungskosten eines Klassifikationssystems definieren als:

$$Kosten = \sum_{i \in T} \sum_{j \in T} P_{i,j} * C_{i,j}$$

Hier stellen die P_{ij} die Wahrscheinlichkeiten einer Entscheidung des Systems dar und die C_{ij} die damit verbundenen Kosten aus der Matrix der Entscheidungskosten. In dem folgenden Ansatz versuchen wir, genau diesen Erwartungswert für die Entscheidungskosten eines fallbasierten Systems zu minimieren. Im folgenden Abschnitt wollen wir darlegen, wie wir diese Anforderung durch eine teilautonome Suche im Entwicklungsgraphen mit geringem Aufwand erfüllen können.

4.3 Die Minimierung der Entscheidungskosten durch das konjugierte Gradientenverfahren

Um ein fallbasiertes System zu finden, das in der Kreditwürdigkeitsprüfung die gestellten Anforderung der Entscheidungskostenbehandlung erfüllt, versuchen wir die Gewichte zur Berechnung der Ähnlichkeit automatisch zu lernen. Wie bereits im Abschnitt 3.3 diskutiert, ist dies ein Verfahren, welches wenig zusätzlichen Akquisitionsaufwand erfordert. Für die Suche im Entwicklungsgraphen verwenden wir einen „Generate-and-Test" Algorithmus, nämlich das konjugierte Gradientenverfahren.

Dieser Algorithmus versucht, schrittweise die Attributgewichte der Ähnlichkeitsberechnung so zu ändern, daß eine gegebene Fehlerfunktion minimiert wird.

Der Basisalgorithmus stellt sich wie folgt dar:

Algorithmus: das konjugierte Gradientenverfahren

1. Initialisiere die Gewichte der Attribute w_a und die Lernrate λ
2. Berechne den Wert der Fehlerfunktion E(w)
3. **While** not(Stop-Kriterium) **Do**
   a) Lernschritt: $\forall$ a $\hat{w}_a := w_a - \frac{\partial E}{\partial w_a} * \lambda$
   b) Berechne: $E(\hat{w})$
   c) **If** $E(\hat{w}) < E(w)$ **Then** $w := \hat{w}$ **Else** $\lambda = \frac{\lambda}{2}$
4. Ausgabe der resultierenden Gewichtsvektoren: w

Nachdem die Gewichte und die Lernrate initialisiert wurden (vgl. Abschnitt 4.3.1), führt der Algorithmus eine Anzahl von Lernschritten durch, bis das Stop-Kriterium erfüllt ist. Im Entwicklungsgraphen werden also eine Menge von Nachfolgerknoten erzeugt bis ein Prototyp entstanden ist, der die Anforderung an die Entscheidungskosten erfüllt. Ist ein Lernschritt erfolgreich, d.h. der Erwartungswert der Kosten geht zurück, dann werden die Gewichte entsprechend modifiziert. Nachdem der Algorithmus terminiert, wird der resultierende Gewichtsvektor als Ergebnis des Lernverfahrens ausgegeben. Die Fehlerfunktion drückt hier einen Fehler in bezug auf die Entscheidungskosten aus. Für die Entscheidungskosten ergibt sich eine Fehlerfunktion aus der Summe der multiplizierten Einträge der Confusionmatrix und der Matrix der Entscheidungskosten unter Beachtung der jeweiligen Vorzeichen für Kosten und Nutzen, also[3]:

$$E := \sum_{q \in FB} \sum_{t \in T} \mathrm{sgn}(C_{tq,t}) * C^2_{tq,t} * p^2_{q,t}$$

Wir wollen nun die Wahl der verschiedenen Parameter des Verfahrens diskutieren. Zu den Parametern, die das Ergebnis des Lernalgorithmus beeinflussen, zählen die initiale Lernrate und das Abbruchkriterium. Durch die Lernrate λ wird die Schrittweite in jedem Lernschritt in Richtung des konjugierten Gradienten gesteuert und das Abruchkriterium gibt an, wann der Algorithmus terminieren soll.

4.3.1 Die Schrittweite im Entwicklungsgraphen und die Wahl der Lernrate

Die Lernrate gibt an, wie groß eine Änderung der Attributgewichte in einem Lernschritt ist. Eine zu kleine Lernrate, also eine kleine Änderung der Gewichte pro Lernschritt, sorgt für lange Laufzeiten des Lernverfahrens. So müssen sehr viele Schritte bei der Navigation durch den Entwicklungsgraphen durchgeführt werden, um einen Gewichtsvektor zu finden, so daß die Anforderungen an das fallbasierte System erfüllt sind. Für den Lernalgorithmus bedeutet dies, daß viele Lernschritte notwendig sind, um zu einem lokalen Minimum der Fehlerfunktion zu gelangen. Wählt man die Lernrate zu groß, wird das Systemverhalten von zwei benachbarten Knoten im Entwicklungsgraphen sehr unterschiedlich. Somit werden Systeme, die durch kleinere Schritte erzeugt würden, bei der Navigation durch den Graphen übergangen.

Dieses Verhalten des Lernalgorithmus ist noch einmal in der Abbildung 2 dargestellt.

Nach der Initialisierung der Attributgewichte durch einen Experten wird mit einer zu großen Lernrate das Minimum min_2 erreicht. Das lokale Minimum in der Umgebung der Vorgabe durch den Experten liegt hier jedoch um den Punkt min_1. Mit einer kleiner gewählten Lernrate würde der Algorithmus dem Pfad mit der durchgezogenen Linie folgen und um den Punkt min_1 terminieren.

4.3.2 Die Anforderungen an das System und die Wahl des Abbruchkriteriums

Das automatische Durchführen von einzelnen Navigationsschritten im Entwicklungsgraphen, hier also das Ausführen von einzelnen Schritten mit unserem Lernalgorithmus, kann beendet werden, falls das fallbasierte System den gestellten Anforderungen genügt. Das Abbruchkriterium des Lernalgorithmus sollte also erfüllt sein, falls durch das Lernen der Gewichte alle Anforderungen, die an das fallbasierte System gestellt werden, erfüllt sind. Die Überprüfung dieser Anforderungen ist jedoch sehr aufwendig. Zum einen werden meistens mehrere unterschiedliche Anforderungen an ein System gestellt (siehe Ab-

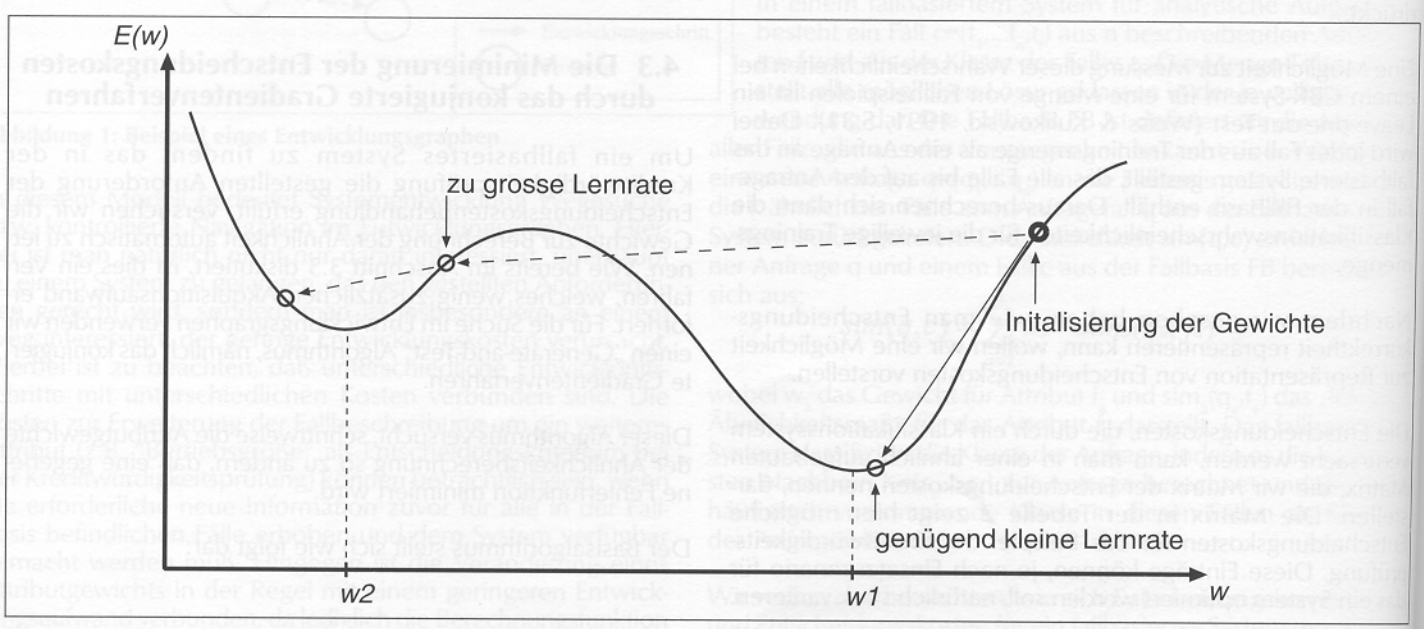


Abbildung 2: Die Wahl der Lernrate und der Einfluß auf das gefundene lokale Minimum

schnitt 3.2) und zum anderen ist die Überprüfung aller Kriterien nach jedem Lernschritt recht aufwendig. Deshalb greift man häufig auf Kriterien zurück, die lokal bei der teilautonomen Suche im Entwicklungsgraphen zur Verfügung stehen und die sich nur auf ein Kriterium, also hier die Entscheidungskosten, beziehen. Mögliche Abbruchkriterien sind hier eine feste Anzahl von Lernschritten, eine minimale Änderung in der Fehlerfunktion oder in den Gewichten oder eine minimale Lernrate.

5. Empirische Bewertung

Ein Vorteil des hier vorgestellten Verfahrens ist, daß es sich je nach Anforderung, die an das fallbasierte System gestellt wird, parametrieren läßt. Sollen andere Kriterien (z.B. Entscheidungskorrekheit) durch das Zielsystem erfüllt werden, so kann das Verfahren durch Verwendung einer anderen stetig differenzierbaren Fehlerfunktion angepaßt werden. Eine Fehlerfunktion zur Minimierung der Entscheidungskosten haben wir hier vorgestellt. Zum Vergleich wurden Experimente zum Lernen von Attributgewichten mit einer Fehlerfunktion durchgeführt, die den Fehler bei der Entscheidungskorrektheit beschreibt. Eine ausführliche Beschreibung dieses Verfahrens befindet sich in (Wilke & Bergmann, 1996). Wir wollen nun diese beiden Lernalgorithmen in einer realen Anwendung aus dem Bereich der Kreditwürdigkeitsprüfung, die an der Universität Kaiserslautern durchgeführt wurde, untersuchen. Die beiden Lernalgorithmen wurden als Erweiterung des fallbasierten Entwicklungswerkzeugs INRECA (Althoff, Auriol et al., 1995) realisiert.

5.1 Die Eigenschaften der Testdaten aus der Kreditwürdigkeitsprüfung

Für unsere Experimente haben wir reale Kundendaten von einer Schweizer Großbank aus dem Bereich der Kreditwürdigkeitsprüfung benutzt. Insgesamt verfügten wir über 685 Fälle. Jeder Fall hatte 136 verschiedene Attribute und eine Klassenbeschreibung. Zwanzig Attribute waren numerisch, während die restlichen Attribute symbolische Werte enthielten. Ungefähr fünf Prozent der Attribute in den Fällen waren unbekannt. In einem Testlauf haben wir 70% der Fälle als Trainingsmenge zum Lernen der Attributgewichte benutzt und die restlichen 30% zum Überprüfen unserer Lernergebnisse. Zum Beginn eines Testlaufs wurden die Attributgewichte zufällig initialisiert und die Fallbasis bestand aus einer zufällig gewählten Trainingsmenge. Mit diesem so aufgebauten fallbasierten System[4] haben wir dann Gewichte mit den beiden vorgestellten Algorithmen gelernt und mit der jeweiligen Testmenge aus den restlichen Fällen die Ergebnisse des Lernens überprüft.

5.2 Die Ergebnisse der empirischen Evaluierung der Algorithmen

Wir wollen nun die Entscheidungskosten der initialen fallbasierten Systeme mit den Entscheidungskosten der Systeme nach dem Lernen der Attributgewichte vergleichen. Am Beispiel der initialen Systeme wollen wir nun zeigen, wie sich die Entscheidungskosten für ein fallbasiertes System berechnen. Dazu betrachten wir die Tabelle 3. Ein Eintrag in dem linken Teil der Tabelle gibt die durchschnittliche Anzahl von Fällen mit entsprechendem Klassifikationsausgang an, wobei eine Grundgesamtheit von 206 Fällen (30%) zum Test vorlag. Auf der rechten Seite befinden sich die durch die jeweiligen Klassifikationen entstehenden Kosten. Die Entscheidungskosten der fallbasierten Systeme ergeben sich dann aus der Summe der einzelnen Kosten für die jeweiligen Entscheidungen und sind unterhalb der Tabelle dargestellt.

	vorhergesagte Klasse vom fallbasierten System			
	Klassifikationsausgänge		Klassifikationskosten	
korrekte Klasse des Falles	**solvent**	**insolvent**	**solvent**	**insolvent**
solvent	***48,6***	***36,4***	***-48,6***	***36,4***
insolvent	***27,4***	***93,6***	***274,0***	***-936,0***
Entscheidungskosten der Systeme: ***-674,2***				

Tabelle 3: Die Berechnung der Entscheidungskosten für die initialen fallbasierten Systeme

Ein negativer Wert bei den Entscheidungskosten repräsentiert hier einen Nutzen, der durch das System entsteht. Wir vergleichen nun diese Entscheidungskosten, mit den Kosten, die die Systeme nach dem Lernen verursachen. Ausgehend von dem initialen System, haben wir die Lernalgorithmen zum Erfüllen der Anforderung Entscheidungskorrektheit und Entscheidungskosten benutzt, um die zufällig initialisierten Gewichte zu verbessern. Die Entscheidungskosten der initialen Systeme und der resultierenden Systeme aus den beiden Lernalgorithmen befinden sich in der Tabelle 4. Die Tabelle zeigt Mittelwerte über fünf unabhängig durchgeführte Testläufe. Man sieht hier, daß beide Lernalgorithmen zu einer Verbesserung der Entscheidungskosten bei den resultierenden Systemen führen. Da wir in unserer Kostenmatrix richtige

	Entscheidungskosten
initiale Systeme ohne Lernen	**-674,2**
Lernen von Gewichten zur Optimierung der Entscheidungskorrektheit	**-756,2**
Lernen von Gewichten zur Optimierung der Entscheidungskosten	**-1040,6**

Tabelle 4: Die gemittelten Entscheidungskosten vor und nach dem Lernen der Attributgewichte

Entscheidungen des Systems mit einem Nutzen und falsche Entscheidungen mit Kosten verbunden haben, ist eine Verbesserung bei den Entscheidungskosten gegenüber den initialen Systemen nicht weiter verwunderlich, obwohl nach dem Kriterium der Entscheidungskorrektheit optimiert wurde. Betrachtet man die Entscheidungskosten der Systeme, die Gewichte zur Verbesserung dieser Kosten benutzt haben, kann man einen weitere Vorteil feststellen. Durch diese Gewichte kam es zwar nicht zu einer Verbesserung der Klassifikationsgüte, aber es wurden die Entscheidungen vom System richtig getroffen, die einen hohen Nutzen erzielen bzw. hohe Kosten vermeiden.

[3] Die Funktion sgn bezeichnet hier die Vorzeichenfunktion. Die Werte der Wahrscheinlichkeiten und der Kosten werden quadriert, um die Fehlerfunktion in eine Energiefunktion zu transformieren.

[4] Die fallbasierten Systeme am Anfang des Lernalgorithmus werden im folgenden als initiale Systeme bezeichnet. Um statistisch relevante Ergebnisse zu erzielen, wurden diese Tests mehrfach durchgeführt und die Ergebnisse gemittelt. Deshalb handelt es sich hier um mehrere initiale Systeme.

Dieser Ansatz sollte auch in anderen Anwendungen, in denen Entscheidungskosten eine wesentliche Rolle spielen, gute Ergebnisse liefern. Die Kosten und der Nutzen, der mit den jeweiligen Entscheidungen verbunden ist, muß bei diesem Verfahren jedoch zusätzlich akquiriert werden.

6. Diskussion und Ausblick

Wir möchten den von uns vorgestellten Ansatz des fallbasierten Schließens zur Kreditwürdigkeitsprüfung mit anderen aus der Literatur bekannten Ansätzen vergleichen. Hierbei wollen wir insbesondere ausnutzen, daß fallbasiertes Schließen zwar zur Klassifikation eingesetzt werden kann (wie in den vorangehenden Kapiteln dargelegt), darüber hinaus aber Freiheitsgrade für die Systementwicklung anbietet, die es erlauben, neben der eigentlichen Klassifikationsaufgabe auch gezielt noch fehlende Information zu erheben (z.B. durch Fragen an den Benutzer). Darüber hinaus können auch Problemfelder bearbeitet werden, wo sowohl numerische als auch qualitative Daten behandelt werden müssen, bei denen strukturierte Fälle wesentlich sind, etc.

Vorteile des fallbasierten Schließens sind z.B.:

- Berücksichtigung konkreter Erfahrungen
- Integration von Problemlösen und Lernen
- Bearbeitung komplexer Problemstellungen mit zahlreichen Ausnahmesituationen
- Behandlung vager Problembeschreibungen
- Integration numerischer (subsymbolischer) und qualitativer (symbolischer) Techniken
- Rechtfertigung von Vorschlägen durch explizit gegebene Fälle
- Einbeziehen generellen Wissens (z.B. in den Phasen „Retrieve" und „Reuse")
- Einbeziehen des gezielten Erhebens fehlender Information
- Vereinfachung – unter gewissen Voraussetzungen – der Entwicklung bzw. Wartung wissensbasierter Systeme.

Allgemeine Voraussetzungen für den Einsatz fallbasierten Schließens sind z.B.:

- Falldaten müssen (elektronisch) verfügbar sein.
- Analytische (direkte) Lösungen sollten nicht auf einfache Weise zu realisieren sein (ansonsten sind diese besser geeignet).
- Fälle müssen bzw. Ähnlichkeit zwischen Fällen muß sinnvoll modellierbar sein.
- Das Fallretrieval muß effizient durchführbar sein.
- Generelles Wissen, das zur Ergänzung des fallbasierten Schließens zusätzlich erforderlich ist, muß verfügbar und repräsentierbar sein.

Wir wollen nun die Grenzen und Probleme anderer, bereits zur Kreditwürdigkeitsprüfung eingesetzter Methoden aufzeigen und fallbasiertes Schließen als eine mögliche Alternative vorschlagen.

Hauschildt (1988a) vertritt die Meinung, daß die Krisendiagnose durch Informationen zu überprüfen ist, die sich nicht in der Bilanz finden. Dies wird bestätigt durch Gemünden (1988), der darauf verweist, daß die Insolvenzprognose auf Basis von Jahresabschlußinformationen nicht unumstritten geblieben und eine Ergänzung durch zusätzliche zukunftsorientierte Informationen sicherlich geboten sei.

Hieraus ergibt sich, daß eine größere Vielfalt an Informationen, z.B. eine Kombination qualitativer, numerischer sowie strukturierter Informationen zu berücksichtigen sind, die vielen klassischen Klassifikationsverfahren Probleme bereiten und für mehr wissensbasierte Ansätze wie z.B. den des fallbasierten Schließens sprechen.

Schwarze & Rosenhagen (1993) argumentieren hier so, daß wissensbasierte Systeme im Gegensatz zu Scoring-Systemen nicht lediglich einen Punktwert als Ergebnis liefern, sondern individuelle Merkmalskonstellationen differenziert berücksichtigen und analysieren können.

Dies wird unterstützt von der Bewertung von Heno (1983, S. 185), in der dieser feststellt, daß die von ihm betrachteten verteilungsfreien Klassifikationsverfahren überwiegend auf geometrischen Ähnlichkeitskonzepten basieren und eine Minimierung der Fehlklassifikationsrate anstreben.

Auch Hauschildts Hinweis (1988c, S. 202), daß ein Diagnosesystem für Unternehmenskrisen ein umfangreiches, systematisch gegliedertes Fragekonzept sei, unterstützt die Tendenz in Richtung eines wissensbasierten Ansatzes, der die gezielte Erhebung fehlender Informationen mit berücksichtigt. Die von Hauschildt (1988c, S. 236ff) beschriebenen Fallbeispiele deuten auch hier an, daß fallbasiertes Schließen eine Möglichkeit sein könnte, einen wissensbasierten Ansatz zur Diagnose von Unternehmenskrisen zu realisieren. Hauschildt (1988c, S. 237) warnt zudem vor einer allzu stark vereinfachenden Schematisierung, weil jeder Fall „anders liege", und verweist in Hauschildt (1988b, S. 16) darauf, daß kein Fall dem anderen gleiche, sondern sich die Fälle allenfalls ähnlich wären.

Mit Blick auf neuronale Netze verweisen Günther & Scheipers (1993) darauf, daß diese interessante Chancen eröffnen, als „Black-Box" aber dem Anwender nicht so transparent seien wie z.B. eine Diskriminanzfunktion. Erste Ansätze würden daher neuronale Netze mit wissensbasierten Systemen verbinden und somit die entsprechenden Entscheidungsregeln und ökonomischen Zusammenhänge aufdecken. Fallbasiertes Schließen bietet durch die Kombinationsmöglichkeit von numerischen und qualitativen Methoden eine Reihe von Möglichkeiten, Techniken aus dem Bereich der künstlichen neuronalen Netze zu integrieren (Althoff, 1996).

Zusammenfassend wollen wir festhalten, daß es eine Reihe von Klassifikationsverfahren gibt, die bislang für die Kreditwürdigkeitsprüfung eingesetzt wurden und daß fallbasiertes Schließen sich in die Reihe dieser Verfahren einordnet. Eine mögliche Vorgehensweise hierzu, die auf dem neuen Ansatz der Minimierung der Entscheidungskosten – im Gegensatz zu den in der Literatur üblichen Ansätzen zur Optimierung der Entscheidungskorrektheit (vgl. z.B. obige Ausführungen zu Heno, 1983) – basiert, wurde in dieser Arbeit vorgestellt. Mit Schwarze & Rosenhagen (1993) sind wir der Meinung, daß Kreditwürdigkeitsprüfungen im Regelfall komplexe und nicht gut strukturierte Probleme sind. Es bietet sich daher der Einsatz wissensbasierter Methoden bzw. ihrer Kombination mit Scoring-Verfahren an. Faßt man die Kreditwürdigkeitsprüfung nicht nur als ein Klassifikationsproblem auf, sondern als eine allgemeine Entscheidungsunterstützungsaufgabe (wie am Beispiel des fallbasierten Schließens in Althoff & Bartsch-Spörl, 1996, bzw. Althoff, 1996, dargelegt), d.h. man betrachtet das Problemfeld entsprechend weit (wie oben z.B. von Hauschildt vorgeschlagen), dann ist fallbasiertes Schließen sicherlich eine interessante Alternative zu künstlichen neuronalen Netzen und klassischen Verfahren.

7. Danksagung

Diese Arbeit wurde unterstützt durch die Kommission der Europäischen Gemeinschaften im Rahmen der Förderung der Projekte INRECA und INRECA-II[5] sowie durch das Wirtschaftsministerium des Landes Rheinland-Pfalz (Stiftung „Rheinland-Pfalz für Innovation") durch die Förderung des Projektes WiMo. Für die Implementierung der vorgestellten Verfahren sowie für viele anregende Diskussionen bedanken wir uns bei Harald Holz und Ivo Vollrath.

[5] INRECA-II (Esprit-Projekt Nr. 22196: „Information and Knowledge Re-Engineering for Reasoning from Cases") mit den Partnern: AcknoSoft (Paris, Koordinator), Irish Medical Systems (Dublin), tecInno (Kaiserslautern), Daimler-Benz AG (Ulm) und der Universität Kaiserslautern.

Literatur

Aamodt, A. & Plaza, E. (1994). Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *AI Comunications*, Vol. 7, No. 1, 39-59.

Althoff, K.-D. (1996). *Evaluating Case-Based Reasoning: The INRECA Case Study*. Habilitationsschrift, Universität Kaiserslautern (eingereicht).

Althoff, K.-D. & Aamodt, A. (1996). Zur Analyse fallbasierter Problemlöse- und Lernmethoden in Abhängigkeit von Charakteristika gegebener Aufgabenstellungen und Anwendungsdomänen. *KI 1/96, Themenheft Fallbasiertes Schließen* (herausgegeben von B. Bartsch-Spörl & S. Wess), 10-15.

Althoff, K.-D., Auriol, E., Barletta, R. & Manago, M. (1995). *A Review of Industrial Case-Based Reasoning Tools*. AI Intelligence, Oxford.

Althoff, K.-D., Auriol, E., Bergmann, R., Breen, S., Dittrich, S., Johnston, R., Manago, M., Traphöner, R. & Wess, S. (1995). Case-Based Reasoning for Decision Support and Diagnostic Problem Solving: The INRECA Approach. In: B. Bartsch-Spörl, D. Janetzko & S. Wess (Hrsg.), Fallbasiertes Schließen - Grundlagen & Anwendungen, 3. Workshop der GI-Fachgruppe „Fallbasiertes Schließen", LSA-Report 95-02, Universität Kaiserslautern

Althoff, K.-D. & Bartsch-Spörl, B. (1996). Decision Support for Case-Based Applications. *Wirtschaftsinformatik 1/96, Schwerpunktthema Fallbasierte Entscheidungsunterstützung* (herausgegeben von D. Ehrenberg), 8-16.

Althoff, K.-D., Richter, M. M. & Wilke, W. (1996). Case-Based Reasoning – A New Technology for Experience Based Construction of Knowledge Systems. (*in Vorbereitung*).

Baetge, J. & Krause, C. (1994). Der Einsatz Künstlicher Neuronaler Netze zur Kreditwürdigkeitsprüfung. In: B. Schiemenz (Hrsg.), *Interaktion – Modellierung, Kommunikation und Lenkung in komplexen Organisationen*, Duncker & Humblot Verlag, 31-54.

Bartsch-Spörl, B. (1996). Towards a Methodology for How to Make CBR-Systems Work in Practice. In: H.-D. Burkhard & M. Lenz (Hrsg.) *4th German Workshop on Case-Based Reasoning - System Development and Evaluation* -, Informatik-Berichte Humboldt Universität Berlin, 36-42.

Bartsch-Spörl, B. & Wess, S. (Hrsg.) (1996). Themenheft Fallbasiertes Schließen. *KI 1/96*, Scientec Publishing.

Bergmann, R., Munoz, H. & Veloso, M. (1996). Fallbasiertes Planen: Ausgewählte Methoden und Systeme. *KI 1/96, Themenheft Fallbasiertes Schließen* (herausgegeben von B. Bartsch-Spörl & S. Wess), 22-28.

Berndt, M. (1995). Einsatz Neuronaler Netze in der Finanzprognose und -analyse. In: W. König (Hrsg.), *Tagungsband Wirtschaftsinformatik ,95*, Physica Verlag, 329-340.

Beuter, H. B., Reiss, I. & Rust, H. J. (1994). Erfahrungen mit formalisierten Verfahren bei der Kreditwürdigkeitsprüfung. In: B. Schiemenz (Hrsg.), *Interaktion – Modellierung, Kommunikation und Lenkung in komplexen Organisationen*, Duncker & Humblot Verlag, 55-74.

Ehrenberg, D. (Hrsg.) (1996). Schwerpunktthema Fallbasierte Entscheidungsunterstützung. *Wirtschaftsinformatik 1/96*, Vieweg Verlag.

Gemünden, H. G. (1988). Defizite der empirischen Insolvenzprognose. In: J. Hauschildt (Hrsg.), *Krisendiagnose durch Bilanzanalyse*, Köln: Verlag Dr. Otto Schmidt, 135-152.

Günther, T. & Scheipers, T. (1993). Ergebnisse der empirischen Insolvenzprognoseforschung auf Basis von Jahresabschlußinformationen. *DStR 29/93*, 1077-1083.

Hauschildt, J. (1988a). Vorwort. In: J. Hauschildt (Hrsg.), *Krisendiagnose durch Bilanzanalyse*, Köln: Verlag Dr. Otto Schmidt, v-vi.

Hauschildt, J. (1988b). Unternehmenskrisen – Herausforderungen an die Bilanzanalyse. In: J. Hauschildt (Hrsg.), *Krisendiagnose durch Bilanzanalyse*, Köln: Verlag Dr. Otto Schmidt, 1-16.

Hauschildt, J. (1988c). Überlegungen zu einem Diagnosesystem für Unternehmenskrisen. In: J. Hauschildt (Hrsg.), *Krisendiagnose durch Bilanzanalyse*, Köln: Verlag Dr. Otto Schmidt, 200-242.

Heno, R. (1983). *Kreditwürdigkeitsprüfung mit Hilfe von Verfahren der Mustererkennung*. Bern, Stuttgart: Verlag Paul Haupt.

Manago, M., Althoff, K.-D., Auriol, E., Bergmann, R., Breen, S., Richter, M. M., Stehr, J., Traphöner, R. and Wilke, W. (1995). *INRECA II: Information and Knowledge Reengineering for Reasoning from Cases*. Projektantrag, Esprit IV (RTD).

Mertens, P., Borkowski, V. & Geis, W. (1993). *Betriebliche Expertensystemanwendungen*. Springer Verlag.

Michie, D., Spiegelhalter, D. J. &. Taylor, C.C. (Hrsg.) (1994). *Machine Learning, Neural and Statistical Classification*, Ellis Horwood.

Oxmann, R. & Voss, A. (1996). Fallgestütztes Entwerfen. *KI 1/96, Themenheft Fallbasiertes Schließen* (herausgegeben von B. Bartsch-Spörl & S. Wess), 29-35.

Reinartz, T. & Wilke, W. (1995). Fallbasiertes Schließen in der Finanzwelt: Eine echte Alternative zu Neuronalen Netzen? In: G. Bol, G. Nakhaeizadeh & K.-H. Vollmer (Hrsg.), *Finanzmarktanalyse und -prognose mit innovativen quantitativen Verfahren*, Physica-Verlag, 15-34.

Richter, M. M. (1995). The Knowledge Contained in Similarity Measures. Invited talk at the First International Conference on CBR (ICCBR-95). Slide copies and an abstract with personal remarks available at: URL: http://wwwagr.informatik.uni-kl.de/CBR/CBR-Homepage/.

Rösler, J. (1988). Die Entwicklung der statistischen Insolvenzdiagnose. In: J. Hauschildt (Hrsg.), *Krisendiagnose durch Bilanzanalyse*, Köln: Verlag Dr. Otto Schmidt, 102-114.

Schwarze, J. & Rosenhagen, K. (1993). Expertensysteme in der Kreditwürdigkeitsprüfung. *WiSt Heft 6 Juni 1993*, 291-295.

Voss, A., Bartsch-Spörl, B. & Oxman, R. (1996). A Study of Case Adaptation Systems. In: Gero, J. S. & Sudweeks, F.(Hrsg.), *Artificial Intelligence in Design '96*, Kluwer Academic Publishers, 173-189.

Weiss, Sh. M. & Kulikowski, C. A. (1991). Computer Systems That Learn - Classification and Prediction Methods from Statistics, Neural Nets, *Machine Learning and Expert Systems*. Morgan Kaufman.

Wess, S. (1995). *Fallbasiertes Schließen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*. Dissertation, Universität Kaiserslautern; auch: infix Verlag.

Wilke, W. & Bergmann, R. (1996). Considering Decision Cost During Learning of Feature Weights, *European Workshop on Case-Based Reasoning* (EWCBR´96), Springer.