CreditCBR: Fallbasierte Entscheidungsunterstützung mit INRECA

Wolfgang Wilke
Universität Kaiserslautern
Fachbereich Informatik, AG Richter
Zentrum für lernende Systeme und Anwendungen (LSA)
Postfach 3049
67653 Kaiserslautern
e-Mail: wilke@informatik.uni-kl.de

Zusammenfassung

Fallbasiertes Schließen (engl.: Case-based Reasoning) hat in den vergangenen Jahren zunehmende Bedeutung für den praktischen Einsatz in realen Anwendungsbereichen erlangt. In dieser Arbeit werden zunächst die allgemeine Vorgehensweise und die verschiedenen Teilaufgaben des fallbasierten Schließens vorgestellt. Anschließend wird auf die charakteristischen Eigenschaften eines Anwendungsbereiches eingegangen und an der konkreten Aufgabe der Kreditwürdigkeitsprüfung die Realisierung eines fallbasierten Ansatzes in der Finanzwelt beschrieben.

1 Was ist fallbasiertes Schließen

Der Mensch lernt aus Erfahrungen. Mit jeder neuen Erfahrung wächst sein Wissen und die Möglichkeit, sich in Zukunft in ähnlichen Situationen durch Erinnerung an eine entsprechende Erfahrung angemessen zu verhalten. Diese Vorgehensweise stellt das Prinzip des *analogen Schließens* dar [4]. In Abbildung 1 wird dieses Prinzip illustriert. Bei einem gegebenen aktuellen Problem versucht man sich zunächst an eine ähnliche bekannte Problemstellung zu erinnern. Anschließend wird die dazu bekannte Lösung auf die aktuelle Situation übertragen.
Fallbasiertes Schließen (engl.: *Case-based reasoning*) stellt ein Modell dieser menschlichen Schlußweise dar, das Problemlösen, Verstehen und Lernen in Gedächtnisprozessen integriert [7]. Die wesentliche Voraussetzung für den Einsatz von fallbasiertem Schließen ist eine geeignete Auswahl und Darstellung bereits bekannter Erfahrungen. Jede dieser Erfahrungen wird als ein *Fall* in einer sogenannten *Fallbasis* gespeichert.

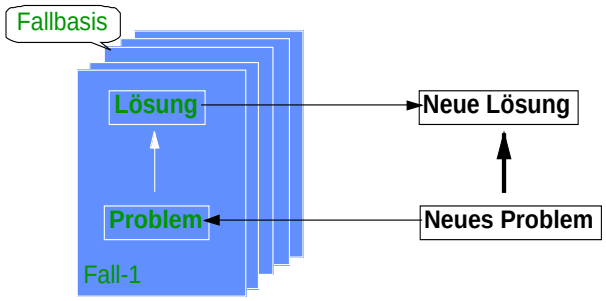

Abbildung 1: Prinzip des analogen Schließens

Jeder Fall besteht mindestens aus einer *Problembeschreibung* und einer Darstellung der entsprechenden *Lösung*. Darüber hinaus wird häufig Information zu dem Lösungsweg oder fehlgeschlagenen Lösungsversuchen in einem Fall verwahrt [8]. Die Lösung eines aktuellen Problems beginnt daraufhin mit einer geeigneten Formulierung der Problemsituation, die dann mit den bereits bekannten Erfahrungen verglichen wird. Die bereits bekannte Lösung wird anschließend, in gegebenenfalls modifizierter Form, auf die aktuelle Situation übertragen. Somit ist die neue Problemstellung durch die bereits gemachten Erfahrungen gelöst. Auf diese Weise entsteht ein neuer Fall, der ebenfalls in der Fallbasis abgelegt werden kann, um so für zukünftige Problemstellungen zur Verfügung zu stehen.
Im folgenden wird zunächst die Technik des fallbasierten Schließens aufgrund eines von Aamodt und Plaza [1] vorgeschlagenen Prozeßmodells vorgestellt. Anschließend werden die Einsatzmöglichkeiten des fallbasierten Schließens in der Finanzwelt am Beispiel der Kreditwürdigkeitsprüfung diskutiert und ein konkreten Lösungsansatz vorgestellt, der an der Universität Kaiserslautern durchgeführt wurde.

2 Ein Prozeßmodell nach Aamodt und Plaza

Abbildung 2 zeigt das Prozeßmodell des fallbasierten Schließens [1]. Im Mittelpunkt dieses Modells steht die *Fallbasis*, die eine Menge ausgewählter Fälle enthält. Darüber hinaus kann zusätzliches *Wissen* aus dem entsprechenden Anwendungsbereich nützlich sein. Solches Wissen dient beispielsweise der Unterstützung der Suche nach einer geeigneten ähnlichen Erfahrung oder kann bei einer gegebenenfalls notwendigen Anpassung einer bekannten Lösung an eine aktuelle Situation verwendet werden. Dieses Wissen ist i.d.R. allgemeinerer Natur als die einzelnen konkreten Erfahrungen einer Fallbasis. Der eigentliche Prozeß des

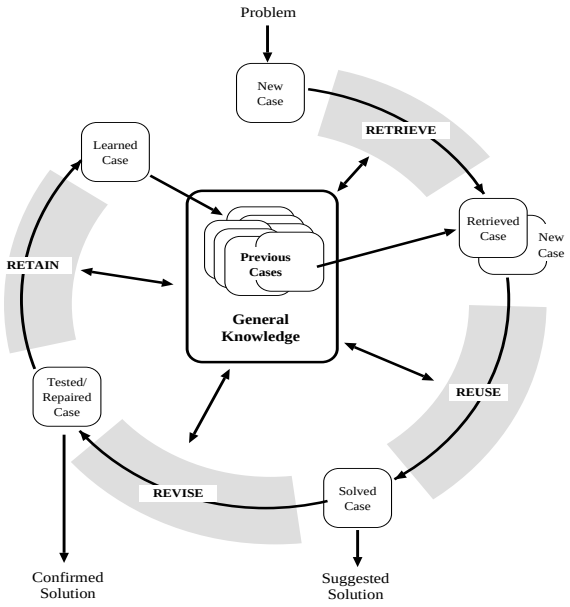

Abbildung 2: Der CBR-Zyklus nach [1]

fallbasierten Problemlösens beginnt mit der Beschreibung des aktuellen *Problems*, das einen *neuen Fall* - allerdings ohne Lösung - definiert. Anschließend folgen vier Schritte:

- *Retrieve*:
  Der erste Schritt des Prozeßmodells dient der Suche nach einem geeigneten Kandidaten für die Wiederverwendung in der aktuellen Situation. Ein Kandidat ist insbesondere dann geeignet, wenn er eine hinreichend ähnliche Problemstellung hat.

- *Reuse*:
  Der nächste Schritt besteht nun in der Übertragung der im ersten Schritt gefundenen Lösung auf die aktuelle Situation. Hier werden grob zwei unterschiedliche Prinzipien unterschieden: Die Kopie der bekannten Lösung oder die Anpassung der bekannten Erfahrung an die neue Situation.

- *Revise*:
  Die erzeugte Problemlösung für die aktuelle Situation ist nicht notwendigerweise korrekt. Aus diesem Grund wird die vorgeschlagene Lösung im dritten Schritt des fallbasierten Modells überprüft und gegebenenfalls korrigiert.

- *Retain*:
  Im letzten Schritt wird schließlich die neu erzeugte Erfahrung für zukünftiges Problemlösen zur Verfügung gestellt. Dabei muß zunächst entschieden werden, ob der neue Fall nützliche Information enthält, die zur Lösung zukünftiger Probleme beitragen. Ist dies der Fall, wird die Problemstellung zusammen mit ihrer Lösung in der Fallbasis abgelegt.

Nachdem die einzelnen Schritte und Teilaufgaben des fallbasierten Schließens im allgemeinen vorgestellt wurden, wird nun auf eine konkrete Anwendung eingangen, die Kreditwürdigkeitsprüfung. Es wird ein fallbasierter Ansatz beschrieben, der in dem Projekt CREDITCBR verfolgt wurde.

3 Ein Anwendungsbeispiel: Die Kreditwürdigkeitsprüfung

Die Kreditwürdigkeitsprüfung ist eine typische Klassifikationsaufgabe. Aufgrund verschiedener Merkmale eines potentiellen Kreditnehmers erfolgt eine Bewertung des Kunden hinsichtlich seiner Kreditwürdigkeit. Der Kunde muß demzufolge einer der beiden Klassen *kreditwürdig* oder *nicht kreditwürdig* zugeordnet werden. In der Situation, in der ein potentieller Kunde ein Wirtschaftsunternehmen ist, gibt es also die beiden möglichen Klassen *solvent* oder *insolvent*. Während in Klassifikationsaufgaben die *Entscheidungskorrektheit* im Vordergrund steht, bietet es sich in dieser Anwendung an, die *Entscheidungskosten* (also die Kosten, die durch Entscheidungen des Systems verursacht werden) zu betrachten und zu optimieren. Zunächst wird jedoch ein fallbasiertes System zur Lösung von Klassifikationsaufgaben etwas genauer beschreiben.

3.1 Fallbasiertes Klassifizieren mit den k nächsten Nachbarn

In einem fallbasiertem System für analytische Aufgaben besteht ein Fall $c = (f_1, \ldots, f_n, t_f)$ aus n beschrei-

benden Attributen f_i und aus der Klasse des Falles t_f. 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:

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

wobei w_a das Gewicht für Attribut f_a und $sim_a(q_a, f_a)$ das lokale Änlichkeitsmaß für das Attribut f_a darstellt. Die Gewichte bestimmen hier, in welchem Verhältnis die einzelnen Attribute in die Berechnung der Ähnlichkeit zwischen Fällen und der Anfrage eingehen. Das fallbasierte System klassifiziert die Klasse der Anfrage, indem es die k nächsten Nachbarn zur Anfrage berechnet und die am häufigsten vorkommende Klasse in diesen Fällen als Klasse des Anfragefalls wählt. Somit haben die Gewichte einen wesentlichen Einfluß auf die Fälle, die als ähnlich zur Problemstellung erkannt werden und somit auf die vorgeschlagene Lösung des Systems.
Im folgenden Abschnitt wird nun auf die Repräsentation von Entscheidungskorrektheit und Entscheidungskosten am Beispiel der Kreditwürdigkeitsprüfung näher eingegangen, um dann darauf aufbauend einen Algorithmus zur Minimierung der Entscheidungskosten von fallbasierten Systemen vorzustellen.

3.2 Repräsentation von Klassifikationsgüte und Kosten

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 [9, 6], wie in Tabelle 1 dargestellt. Hier bezeichnen die einzelnen Einträge $P_{i,j}$ die Wahrscheinlichkeit, daß für eine Anfrage der Klasse i vom System die Klasse j als Lösung vorhergesagt wird. Diese Wahrscheinlichkeiten können aus einer Menge von bereits bekannten Fällen berechnet werden.

$P_{i,j}$	*vorhergesagte Klasse*	
richtige Klasse	**solvent**	**insolvent**
solvent	0.275	0.225
insolvent	0.125	0.375

Tabelle 1: Ein Beispiel für eine Confusionmatrix in der Kreditwürdigkeitsprüfung

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 viel größere Rolle als die Entscheidungskorrektheit des Systems. Kosten sind hierbei solche, die dem Kreditinstitut aufgrund einer falschen Entscheidung entstehen (Verlust des Kreditvolumens bzw. Verzicht auf den Zinsertrag). Nutzen ist hier der Gewinn für das Unternehmen, der durch richtige Entscheidungen entsteht (z.B. Zinserträge bei Kreditgeschäften mit einem solventen Kunden). In diesem Fall besteht folglich das Ziel der Entwicklung eines Systems darin, ein System zu erhalten, das möglichst geringe Entscheidungskosten verursacht. Die Entscheidungskosten in dieser Anwendung sind je nach Anfragefall und der Bewertung durch das System sehr unterschiedlich. Wird ein insolventer Kunde vom System als solvent eingestuft, so ist es sehr wahrscheinlich, daß ein Großteil der gesamten Kreditsumme der Bank verloren geht (hohe Kosten). Wird dahingegen ein solventer Kunde als insolvent vom System klassifiziert, verliert die Bank nur die Zinserträge, die sie bekommen hätte, falls das Kreditgeschäft zustande gekommen wäre (geringe Kosten). Somit sind die Kosten, die durch Entscheidungen in dieser Anwendung entstehen können, sehr unterschiedlich. Um nun diese Entscheidungskosten darzustellen, kann man ebenfalls eine Matrix verwenden, *die Matrix der Entscheidungskosten*, wie sie beispielhaft in Tabelle 2 gezeigt ist. Die Einträge $C_{i,j}$ bezeichnen hier die Kosten, die durch eine Klassifikation eines Anfragefalles i als Klasse j durch das System entstehen.
Mit der Confusionmatrix und der Matrix der Entscheidungskosten können wir nun den *Erwartungswert für die gesamten Entscheidungskosten* eines Klassifi-

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

Tabelle 2: Ein Beispiel für eine Matrix der Entscheidungskosten aus der Kreditwürdigkeitsprüfung

kationssystems definieren als:

$$Cost_{decision} = \sum_{i \in T} \sum_{j \in T} P_{i,j} * C_{i,j}, \qquad (2)$$

wobei T die Menge aller möglichen Klassen, also hier *solvent* und *insolvent* bezeichnet. Im folgenden Ansatz wird genau dieser Erwartungswert für die Entscheidungskosten eines fallbasierten Systems minimiert.

3.3 Das konjugierte Gradientenverfahren zur Bestimmung der Attributgewichte

Um ein fallbasiertes System zu finden, das in der Kreditwürdigkeitsprüfung die notwendige Anforderung erfüllt, also minimale Entscheidungskosten erzielt, versucht der folgende Ansatz die Gewichte w_a zur Berechnung der Ähnlichkeit automatisch zu lernen. Dafür wird ein Generate-and-Test Algorithmus verwandt, 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:

1. Initalisiere den Gewichtsvektor $w = (w_1, \ldots, w_n)$ und die Lernrate λ
2. Berechne die Fehlerfunktion $E(w)$ des anfänglichen fallbasierten System
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 des Gewichtsvektors w

Zuerst werden die Gewichte und die Lernrate initialisiert. Die Gewichte können zufällig oder durch einen Experten initialisiert werden. Die Lernrate gibt an, wie groß eine Änderung der Attributgewichte in einem Lernschritt ist. Nachdem der initiale Wert der Fehlerfunktion berechnet wurde, führt der Algorithmus eine Anzahl von Lernschritten durch, bis das Stop-Kriterium erfüllt ist. 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 folgende Fehlerfunktion:

$$E_{cost} = \sum_{q \in FB} \sum_{t \in T} sgn(C_{t_q,t}) * C^2_{t_q,t} * p^2_{q,t} \qquad (3)$$

Der Fehler berechnet sich also aus der Summe der multiplizierten Einträge der Confusionmatrix und der Matrix der Entscheidungskosten unter Beachtung der jeweiligen Vorzeichen für Kosten und Nutzen. Diese Fehlerfunktion gilt es also mit dem konjugierten Gradientenverfahren zu minimieren, um zu minimalen Entscheidungskosten des fallbasierten Klassifikationssystems zu gelangen.
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 wurde 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 [11], die auf den Arbeiten von Lowe [5] und Wettschereck und Aha [10] aufbaut.

3.4 Ergebnisse der Tests

Es werden nun die Lernalgorithmen zur Verbesserung der Entscheidungskorrektheit und der Entscheidungskosten in einer realen Anwendung aus dem Bereich der Kreditwürdigkeitsprüfung, die an der Universität Kaiserslautern durchgeführt wurde, untersucht. Die beiden Algorithmen wurden als Erweiterung des fallbasierten Entwicklungswerkzeugs INRECA[1] realisiert.
Für die Experimente wurden reale Kundendaten von einer Schweizer Großbank aus dem Bereich der Kreditwürdigkeitsprüfung benutzt. Insgesamt wurden 685

[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.

Fälle für die Tests verwandt. 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 wurden 70% der Fälle als Trainingsmenge zum Lernen der Attributgewichte benutzt und die restlichen 30% zum Überprüfen der Lernergebnisse. Zum Beginn eines Testlaufs wurden die Attributgewichte zufällig initialisiert und die Fallbasis bestand aus einer zufällig gewählten Trainingsmenge. Mit einem so aufgebauten fallbasierten System (Es wird dieses fallbasierte System im weiteren als initiales fallbasiertes System bezeichnet) wurden dann Gewichte zur Verbesserung der Entscheidungskorrektheit und der Entscheidungskosten gelernt und mit der jeweiligen Testmenge aus den restlichen Fällen die Ergebnisse des Lernens überprüft.
Es werden nun die Entscheidungskosten der initialen fallbasierten Systeme[2] mit den Entscheidungskosten der Systeme nach dem Lernen der Attributgewichte verglichen. Am Beispiel der initialen Systeme wird nun gezeigt, wie sich die Entscheidungskosten für ein fallbasiertes System berechnet. Ein Eintrag in der Tabelle 3 gibt die durchschnittliche Anzahl von Fällen mit entsprechendem Klassifikationsausgang an, wobei eine Grundgesamtheit von 206 Fällen (30%) zum Test vorlag. In Tabelle 4 befinden sich die durch die jeweiligen Klassifikationen entstehenden Kosten. Die Klassifikationsausgänge aus der Tabelle 3 wurden hier mit den jeweiligen Kosten aus der Tabelle 2 bewertet. Die Entscheidungskosten der fallbasierten Systeme ergeben sich dann aus der Summe der einzelnen Kosten für die jeweiligen Entscheidungen und sind unterhalb von Tabelle 4 dargestellt. Ein negativer Wert bei den Entscheidungskosten repräsentiert hier einen Nutzen, der durch das System entsteht.

	vorhergesagte Klasse	
Klasse des Anfragefalles	**solvent**	**insolvent**
solvent	48.6	36.4
insolvent	27.4	93.6

Tabelle 3: Die Klassifikationsausgänge für die initialen fallbasierten Systeme

	Klassifikationskosten	
Klasse des Anfragefalles	**solvent**	**insolvent**
solvent	-48.6	36.4
insolvent	274.0	-936.0
Entscheidungskosten: **-674.2**		

Tabelle 4: Die Entscheidungskosten für die initialen fallbasierten Systeme

[2]Um statistisch relevante Informationen zu erlangen, wurden mehrere Tests mit verschiedenen initialen Systemen durchgeführt und die Ergebnisse wurden gemittelt.

Diese Entscheidungskosten werden nun mit den Kosten, die die Systeme nach dem Lernen verursachen, verglichen. Ausgehend von dem initialen System wurden die Lernalgorithmen zum Erfüllen der Anforderungen 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 Tabelle 5. 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 in der Kostenmatrix richtige Entscheidungen des Systems mit einem Nutzen und falsche Entscheidungen mit Kosten verbunden sind, 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 weiteren 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.

	Kosten
ohne Lernen:	**-674.2**
Entscheidungskorrektheit:	**-756.2**
Entscheidungskosten:	**-1040.6**

Tabelle 5: Die Entscheidungskosten vor und nach dem Lernen von Attributgewichten

4 Diskussion und Ausblick

In dieser Arbeit wurde die Methode des fallbasierten Schließens eingeführt und ein Prozeßmodell vorgestellt, das die Vorgehensweise beim Problemlösen mit dieser Technik erläutert. Anschließend ist ein konkreter Ansatz beschrieben worden, der das Problem der

Kreditwürdigkeitsprüfung behandelt. Eine Repräsentation von Entscheidungskorrektheit und von Entscheidungskosten wurde eingeführt. Darauf aufbauend ist ein Algorithmus entworfen worden, der den Erwartungswert der Entscheidungskosten durch eine Modifikation der Attributgewichte verbessert. Dieser Algorithmus wurde dann in einer empirischen Evaluation mit realen Kundendaten mit einem Verfahren verglichen, das die Entscheidungskorrektheit eines fallbasierten Systems mit Gewichtslernen optimiert. Eine ausführliche Beschreibung beider Verfahren findet man in [11].
Dieser Ansatz sollte auch in anderen Anwendungen gute Ergebnisse liefern, in denen Entscheidungskosten eine wesentliche Rolle spielen. Die Kosten und der Nutzen, der mit den jeweiligen Entscheidungen verbunden ist, muß bei dem vorgestellten Verfahren jedoch zusätzlich akquiriert werden.
Auch andere Lernalgorithmen zum Verbessern fallbasierter Systeme könnten so modifiziert werden, daß sie den Erwartungswert der Entscheidungskosten betrachten. Ein Instance-based learning Algorithmus [2, 3] versucht zum Beispiel nur die Fälle in der Fallbasis zu belassen, die für gute Klassifikationsergebnisse sorgen. Die Fälle, die falsche Entscheidungen verursachen, werden durch diesen Algorithmus eliminiert. Hierbei werden aber die Fälle aus der Fallbasis nur nach der Anzahl ihrer richtigen und falschen Vorhersagen bewertet. Würde man in diese Bewertung die Entscheidungskosten mit eingehen lassen, könnte man auch die Fallbasis dahingehend optimieren, daß sich der Erwartungswert der Entscheidungskosten verbessert.
Ein Vergleich des vorgestellten Ansatzes des fallbasierten Schließens zur Kreditwürdigkeitsprüfung mit anderen aus der Literatur bekannten Ansätzen ist in [12] zu finden.

Danksagung

Diese Arbeit wurde unterstützt durch die Kommission der Europäischen Gemeinschaften im Rahmen der Förderung der Projekte INRECA und INRECA-II sowie durch das Wirtschaftsministerium des Landes Rheinland-Pfalz (Stiftung Rheinland-Pfalz für Innovation) durch die Förderung des Projektes WIMO.
Für viele anregende Diskussionen und die Implementierung der vorgestellten Verfahren mochte ich mich hiermit bei Ralph Bergmann, Harald Holz und Ivo Vollrath bedanken.

Literatur

[1] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, March 1994.

[2] David W. Aha. Incremental, instance-based learning of independend and graded concepts. In *Proceedings of the 6th international Workshop on Machine Learning*, pages 387–391, 1989.

[3] David W. Aha. Case-Based Learning Algorithms. In *Proceedings: Case-Based Reasoning Workshop (DARPA)*, pages 147–158, Washington, D.C., May 1991. Morgan Kaufman.

[4] Thomas Evans. A heuristic program to solve geometric analogy problems. In *Semantic Information Processing*. MIT Press, 1963.

[5] D. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7:72–85, 1995.

[6] Donald Michie, David J. Spiegelhalter, and Charles C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

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

[8] Manuela Veloso. *Learning by Analogical Reasoning in General Problem Solving*. Phd thesis cmu-cs-92-174, Carnegie Mellon University, August 1992.

[9] Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn – Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, 1991.

[10] Dietrich Wettschereck and David W. Aha. Weighting features. In Manuela Veloso and Agnar Aamodt, editors, *Case-Based Reasoning Research and Development*, pages 347–358. Springer, October 1995.

[11] Wolfgang Wilke and Ralph Bergmann. Considering Decision Costs During Learning of Feature Weights. In *Proccedings of the EWCBR-96*, 1996.

[12] Wolfgang Wilke and Ralph Bergmann and Klaus-Dieter Althoff. Fallbasiertes Schließen zur Kreditwürdigkeitsprüfung. *KI-Themenheft: KI-Methoden in der Finanzwirtschaft*, 4, 1996 (eingereicht).