Maschinelle Ableitung von zeitlichen Abhängigkeitsbeziehungen auf Basis der Analyse von Musterfolgen

Markus Kraft, Andreas Fick, Hubert B. Keller
Institut für Angewandte Informatik
Forschungszentrum Karlsruhe
Tel.: 07247 / 82-5782, Fax.: -5730
E-mail: fick@iai.fzk.de

Abstract. Die automatische Ableitung regelbasierter, für den Menschen verständlicher Verhaltensmodelle aus Zeitreihen stellt als Erweiterung „klassischer" Identifikationsverfahren [5] einen wichtigen Ansatz zur Modellierung des Verhaltens komplexer dynamischer Systeme dar.
Während sich zahlreiche Arbeiten mit der Anpassung bekannter Verfahren zum Lernen von Klassifikatoren, die direkt zur Verarbeitung numerischer Daten geeignet sind, beschäftigen [6] [3] [12], werden Verfahren des Maschinellen Lernens aus den Bereichen des induktiven logischen Programmierens (ILP) oder der Ableitung formaler Sprachen (grammatical inference) nur selten eingesetzt.
Der vorliegende Artikel beschreibt zum einen eine Möglichkeit zum Übergang von einer numerischen zu einer abstrakten, symbolischen Beschreibungsebene, die die Anwendung solcher Verfahren möglich macht.
Zum anderen wird die Anwendung des Multi-Stream Dependencies Detection Algorithmus (MSDD, siehe [8]) zur Erkennung von zeitlichen Abhängigkeitsbeziehungen zwischen Mustern unterschiedlicher Zeitreihen beschrieben. Der Algorithmus wird dabei um die Verarbeitung unscharfer Auftretenszeitpunkte bei Prämissen- und Konklusionsmustern erweitert.
Die prinzipielle Leistungsfähigkeit des Verfahrens wird an einem einfachen Simulationsbeispiel gezeigt.

Keywords. Zeitreihen, Muster, Clustering, MSDD-Algorithmus

1 Einleitung

Bei dem vorliegenden Artikel handelt es sich um die Darstellung der Ergebnisse aus [7]. Er beschäftigt sich mit der automatischen Ableitung regelbasierter, verständlicher Verhaltensmodelle aus Zeitreihen. Diese stellt als Erweiterung „klassischer" Identifikationsverfahren [5] einen wichtigen Ansatz zur Modellierung des Verhaltens komplexer dynamischer Systeme dar und leistet damit indirekt, beispielsweise im Rahmen modellbasierter Regelungskonzepte, einen wesentlichen Beitrag zur Optimierung der Führung solcher Systeme.

Während sich zahlreiche Arbeiten mit der Anpassung bekannter Verfahren zum Lernen von Klassifikatoren, die direkt zur Verarbeitung numerischer Daten geeignet sind, beschäftigen (siehe z.B. [6], [3] oder [12]), werden Verfahren des Maschinellen Lernens aus den Bereichen des induktiven logischen Programmierens (ILP) oder der Ableitung formaler Sprachen (grammatical inference) nur selten eingesetzt. Dies ist nicht nur aus grundsätzlichen Erwägungen heraus bedauerlich, da so eine ganze Palette von Verfahren brach liegt, sondern auch deshalb, weil diese Verfahren teilweise zum Lernen deutlich komplexerer Zusammenhänge in der Lage sind, als dies beispielsweise Entscheidungsbaumalgorithmen wie C4.5 schon aufgrund der gewählten Repräsentationsform leisten können. Die wichtigste Ursache für die Situation dürfte darin liegen, daß diese Verfahren zum Lernen auf eine symbolische Beschreibungsebene angewiesen sind und auch das Lernergebnis nur in Ausnahmefällen direkt auf numerische Daten angewendet werden kann, sondern im allgemeinen interpretiert, bzw. transformiert werden muß.

Nach der Motivation und einem Überblick über das Verfahren geht der vorliegende Artikel zunächst auf eine Möglichkeit zum Übergang von einer numerischen zu einer abstrakten, symbolischen Beschreibungsebene, die die Anwendung solcher Verfahren möglich macht, ein. Hierzu wird zunächst eine Approximation der Zeitreihen durch Muster vorgenommen. In einem anschließenden Clusterungsschritt werden ähnliche Muster zusammengefaßt, so daß die Zeitreihen als symbolische Folgen von Muster-Prototypen beschrieben werden können.

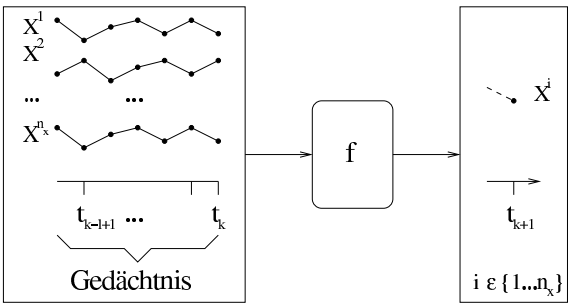
Danach wird die Anwendung des Multi-Stream De-

pendencies Detection Algorithmus (MSDD, siehe [8]) zur Erkennung von zeitlichen Abhängigkeitsbeziehungen zwischen Mustern unterschiedlicher Zeitreihen beschrieben. Der Algorithmus wird dabei um die Verarbeitung unscharfer Auftretenszeitpunkte bei Ursachen- und Wirkungsmustern erweitert.

Die prinzipielle Leistungsfähigkeit des Verfahrens und einige bei der Anwendung aufgetretene Probleme werden an einem einfachen Simulationsbeispiel gezeigt. Der Artikel endet mit einer Zusammenfassung des erreichten Standes des Verfahrens und gibt einen Ausblick auf weitergehende Forschungsaktivitäten.

2 Bezeichnungen und Motivation

Gegeben seien n_x Zeitreihen X^i der Länge n_t zu den äquidistanten Zeitpunkten $t_1, \ldots, t_{n_t}$. Der Abstand aufeinanderfolgender Zeitpunkte sei äquidistant, so daß jeder Zeitpunkt mit seinem Index identifiziert werden kann: $t_j \hat{=} j$. Der gemessene Wert der Zeitreihe X^i, $i \in 1, \ldots, n_x$, zum Zeitpunkt t_j (oder j) sei mit x_j^i bezeichnet. Die Werte aller Zeitreihen zu einem Zeitpunkt t_j seien zu dem Spaltenvektor $x_j = (x_j^1, \ldots, x_j^{n_x})^T$ zusammengefaßt.



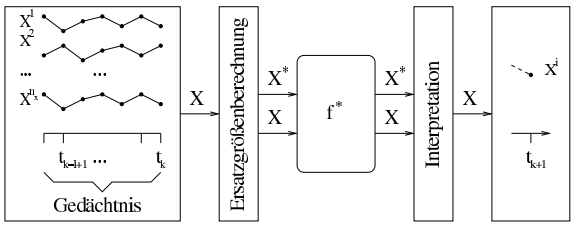
$X^1, \ldots, X^{n_x}$ zusammengefaßten Werten der Ausgangs- und Eingangsgrößen als Abbildungsproblem betrachtet werden. Es ist dann für einen gegebenen Zeitpunkt t_k und eine gewählte Größe X^i aus den Meßwerten innerhalb eines Zeitintervalls der Länge t_l (betrachtete Vergangenheit) der jeweils nächste Wert zu berechnen:

$$\begin{aligned} f &: \mathbb{R}^{l * n_x} \rightarrow \mathbb{R} \\ f(x_{k-l+1}, \ldots, x_k) &= \hat{x}_{k+1}^i. \end{aligned} \tag{1}$$

Aufgabe der Verhaltensmodellierung, bzw. des Lernverfahrens, wäre somit die Identifikation der Funktion f (siehe auch Abbildung 1) zur möglichst guten Berechnung der Schätzwerte $\hat{x}_{k+1}^i$.

Probleme bereiten bei diesem Vorgehen zum einen die Größe des Eingaberaums der Funktion und die im Vergleich dazu, bzw. im Vergleich zur Größe des resultierenden Raums der möglichen Abbildungsfunktionen, geringe Anzahl von Lernbeispielen. Zum anderen ist die oft nicht gegebene Verstämdlichkeit der resultierenden gelernten Funktion problematisch. Neben Maßnahmen der Datenvorverarbeitung, wie z.B. Glätten oder Filtern, haben sich u.a. „Standardmethoden“ wie die Vorauswahl relevanter Größen, die Reduktion der Zeitauflösung, die Verwendung einer geeigneteren Codierung oder die geschickteren Berücksichtigung des Gedächtnisses, die alle auf eine Verkleinerung des Eingangsraumes abzielen, bewährt.

Die Probleme universeller Identifikationsverfahren mit großen Eingaberäumen sowie die Schwierigkeit des Menschen, die resultierenden Modelle zu verstehen, rühren jedoch nicht nur von der Größe der Eingaberäume her. Vielmehr sind sie oft vor allem dadurch begründet, daß diese Räume nicht strukturiert sind. Da jedes Eingabeelement der durch die Größen und die Zeitpunkte gebildeten zweidimensionalen Eingabematrix nach Gleichung 1 gleichbehandelt wird, geht Information über zwischen den Elementen bestehende semantische Zusammenhänge verloren. Dies betrifft beispielsweise die Zugehörigkeit bestimmter Eingabeelemente zu derselben Größe (Zeitreihe) oder die Zugehörigkeit von Eingabeelementen zu demselben Zeitpunkt (Gleichzeitigkeit, Zustand). Insbesondere werden zwischen Eingabeelementen bestehende zeitliche Beziehungen nicht berücksichtigt.



der zu identifizierenden Funktion f^* möglicherweise noch interpretiert werden muß, da auch im Bildbereich von f^* abgeleitete Größen verwendet werden können.

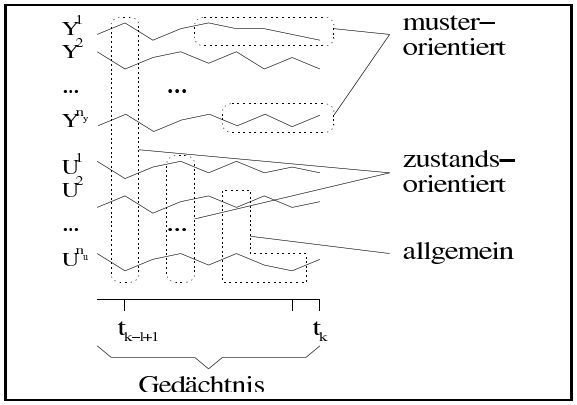


Abbildung 3 Zustands- und musterorientierte Ersatzgrößen

Wie Abbildung 3 verdeutlicht, kann bei abgeleiteten Größen zwischen zustandsorientierten, musterorientierten und allgemeinen Größen unterschieden werden. Erstere entsprechen Projektionen der Meßwertvektoren in einen niedrigdimensionalen Raum. Musterorientierte Größen beschreiben Entwicklungen in einzelnen Zeitreihen, wie z.B. Trends. Im allgemeine Fall können beliebig komplizierte Zusammenhänge repräsentiert werden, bei einem räumlich ausgedehnten System beispielsweise zeitversetzte Auswirkungen von Ereignissen auf verschiedene Meßstellen.

3 Der Ansatz im Überblick

Im folgenden wird ein musterorientierter Ansatz beschrieben, der als Grundlage für eine Verhaltensmodellierung nach Abbildung 2 dienen soll. Dazu werden geeignete Muster definiert und statistisch relevante Abhängigkeiten im zeitlichen Auftreten der Muster verschiedener Zeitreihen erkannt. [1] Ziel ist die Überführung dieser Zusammenhänge in ein regelbasiertes Modell, das zu einer effektiven Prognose in der Lage ist.

Bei dem Ansatz handelt es sich um eine mehrstufiges Verfahren. Zunächst wird eine symbolische Repräsentation der Zeitreihen erstellt. Dies geschieht in den folgenden Verfahrensschritten:

1. Vorverarbeitung (Filterung etc.)
2. Approximation der numerischen Zeitreihen durch Muster
3. Bestimmung von Musterprototypen
4. Symbolische Repräsentation der Zeitreihen

Anschließend wird die eigentlich Suche nach Abhängigkeiten vorgenommen:

5. Definition von Zeitintervallen für das Auftreten von Symbolen bzw. Symbolen
6. Anwendung einer erweiterten Form des MSDD-Algorithmus

Als Ergebnis erhält man eine Liste bewerteter möglicher zeitlicher Abhängigkeitsbeziehungen, die als Regeln interpretierbar sind. Dabei werden sowohl positive (Bei Gültigkeit der Prämisse ist die Wahrscheinlichkeit für das Auftreten der Konklusion erhöht gegenüber dem Fall der Nicht-Gültigkeit der Prämisse.) als auch negative Zusammenhänge (Bei Gültigkeit der Prämisse ist die Wahrscheinlichkeit für das Auftreten der Konklusion erniedrigt.) berücksichtigt.

4 Übergang zu einer symbolischen Repräsentationsebene

Der gesamte Übergang zu einer symbolischen Repräsentationsebene wird interaktiv durch den Benutzer gesteuert, wobei das Verfahren für jeden Arbeitsschritt sinnvolle Ausgangsparameter vorschlägt.

Als Vorverarbeitung sind verschiedene übliche Glättungsalgotithmen (Gaußfilter, gleitendes Mittel, Medianfilter) implementiert worden, die hier nicht näher beschrieben werden sollen.

4.1 Approximation durch Muster

Ziel des Approximationsschrittes ist die Näherung einer Zeitreihe durch eine möglichst geringe Anzahl von Mustern, wobei die Abweichung jedes Musters von der Zeitreihe bestimmten Gütekriterien genügen soll. Als Muster wurden Geradenabschnitte gewählt, da diese sich nicht nur zur schnellen und effizienten Approximation beliebig geformter Kurven eignen, sondern auch intuitiv als „Trends“ interpretierbar sind. Ein Muster ist definiert als Tupel $(s, x_s, e, x_e), s < e$ mit dem Anfangszeitpunkt s, dem Anfangswert x_s, dem Endzeitpunkt e und dem Endwert x_e. Es kann auch als Linie

$$l(j) = x_s + \frac{x_e - x_s}{e - s} * (j - s),\ s \leq j \leq e$$

aufgefaßt werden. Zur Parametrisierung der Approximationsgenauigkeit wurden die zwei Maße

$$distance(l) := \max |l(j) - x_j|,\ s \leq j \leq e$$

und

[1] Genauer gesagt, werden Kombinationen von Mustern gefunden, die in signifikanter Weise häufiger oder weniger häufig aufgetreten sind, als dies bei statistischer Unabhängigkeit zu erwarten gewesen wäre.

$$deviation(l) := \sqrt{\frac{1}{e-s+1}\sum_{j=s}^{e}(l(j)-x_j)^2}$$

gewählt. Ersteres berechnet den maximalen absoluten Abstand des Trends zur Kurve. Letzteres dient dazu, den durchschnittlichen quadratischen Abstand zu bewerten und entspricht einer Schätzung der Standardabweichung. Es wurden verschiedene Varianten von Geradenabschnitten, beispielsweise solche, bei denen Anfangs- und Endpunkt Punkte der Zeitreihe waren, getestet. Die besten Ergebnisse wurden mit einer direkten Minimierung des zweiten Gütemaßes erreicht, was der Berechnung von Regressionsgeraden entspricht. Das erste Gütekriterien bestimmt bei diesem Vorgehen nur Randbedingungen, geht aber nicht direkt in die Berechnung der Muster ein.

Least-Squares Trend-Suche(*timeSeries*, *maxDist*, *maxDev*)

1. *start* = 1
2. while (*start* < LENGTH(*timeSeries*)+1) do
   (a) *trend* = FIND-LS-TREND(*start*, *maxDist*, *maxDev*)
   (b) *end* = END(*trend*)
   (c) if (*end* < LENGTH(*timeSeries*)) then
      i. *next* = FIND-LS-TREND(*end*, *maxDist*, *maxDev*)
      ii. IMPROVE-LS-TREND(*trend*, *next*, *maxDist*, *maxDev*)
   (d) ADD-PATTERN(*trend*)
   (e) *start* = END(*trend*)

Abbildung 4 Algorithmus zur Least-Squares Trend-Suche

Zur Berechnung wurde das in Abbildung 4 skizzierte heuristische Verfahren entwickelt. Der Algorithmus bearbeitet die Zeitreihe von links nach rechts, bis jeder Wert durch ein Muster abgedeckt ist. Zunächst wird ein möglichst langer Trend gesucht, der den Gütekriterien genügt (FIND-LS-TREND $(start, maxDist, maxDev)$). Dies geschieht im wesentlichen durch wiederholte Verlängerung des zu approximierenden Zeitintervals um einen konstanten Faktor und anschließende Regressionsgeradenberechnung. Danach wird der nächstfolgende Trend berechnet (FIND-LS-TREND $(end, maxDist, maxDev)$), und zum Schluß wird die Grenze zwischen beiden Trends im Sinne einer Verbesserung der Gesamtgüte beider Muster verschoben (IMPROVE-LS-TREND $(trend, next, maxDist, maxDev)$). In der folgenden Schleifeniteration wird dann von dieser neu berechneten Grenze aus das nächste Muster berechnet.

Der Algorithmus erwies sich als guter Kompromiß zwischen einer schnellen Berechnung, einer guten Approximationsqualität und einer möglichst geringen Anzahl von resultierenden Mustern. Er stellt sicher, daß jedes gefundene Muster mindestens den Gütekriterien genügt, findet jedoch nicht unbedingt die beste Approximation oder die Approximation mit der geringsten Anzahl von Mustern. Ein Beispiel für die Approximation einer Kurve zeigt Abbildung 5.

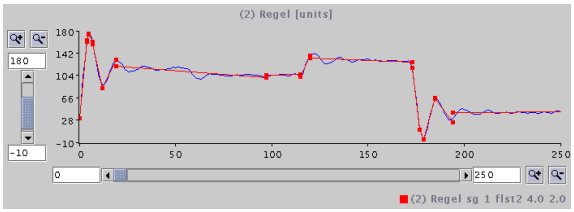


Abbildung 5 Approximation mit der Least-Squares Trend-Suche

4.2 Clustern und Prototypenbildung

Als Ergebnis des Approximationsschrittes erhält man für jede Zeitreihe X^i eine Folge von Mustern, die als neue (kompaktere) Zeitreihe M^i betrachtet werden kann. Zur Zusammenfassung ähnlicher Muster zu Klassen wird als nächstes ein Clusterungs-Schritt vorgenommen. Implementiert wurde u.a. der Fuzzy C-Means Algorithmus [4] [1] [2], der gleichzeitig Clusterzentren berechnet, die als typischer Muster interpretierbar sind.

Abstrahiert man vom absoluten Auftretenszeitpunkt der Muster, so bieten sich folgende Merkmale zum Clustern an:

- Startwert x_s,
- Endwert x_e,
- Dauer $e-s$,
- Werteänderung $x_e - x_s$ und
- Steigung $\frac{x_e-x_s}{e-s}$.

Die sinnvollsten Ergebnisse wurden meist mit einer dreidimensionalen Clusterung nach Startwert, Dauer und Steigung erzielt. Das bedeutet, daß die approximierte Zeitreihe im wesentlichen nach Zustand und Gradient aufgeteilt wird.

Ein mit Daten aus dem sich anschließen Beispiel für eine Clusterzahl $n_c = 5$ erzieltes Resultat zeigt Abbildung 6.

4.3 Generierung symbolischer Zeitreihen

Ordnet man jedem Cluster ein Symbol, z.B. eine Clusternummer c, $c > 0$, zu und ersetzt jedes Muster der Musterfolge M^i durch das Symbol des zugehörenden Clusters, so läßt sich für jede Original-Zeitreihe X^i eine neue symbolische Zeitreihe S^i gleicher Länge erstellen. Für $j = 1, \ldots, n_x$ erhält man:

$$s_j^i = \begin{cases} c(m), \text{ falls } \exists m \in M^i \ : \ \text{START}(m) = t_j \\ 0, \text{ sonst.} \end{cases}$$

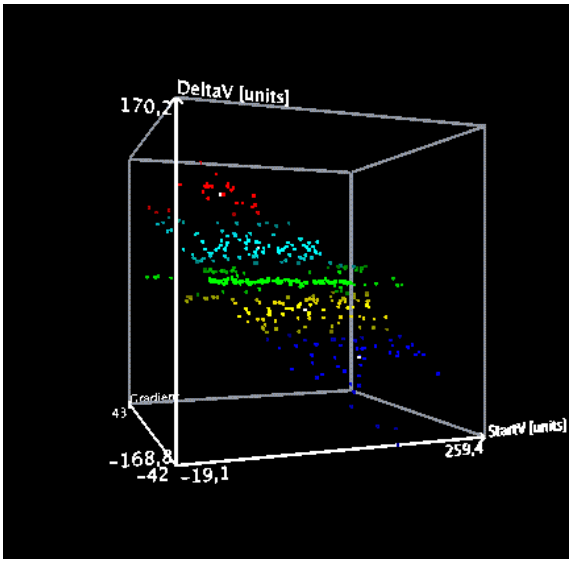


Abbildung 6 Clustern von Mustern

Hierbei bezeichnet $c(m)$ eine Funktion, die die zum Muster m gehörende Nummer berechnet. An Stellen, an denen kein Muster beginnt, wird 0 eingesetzt. Man erhält als Ergebnis eine Zeitreihe, die größtenteils aus Nullen besteht.

Einen Ausschnitt aus so erzeugten symbolischen Zeitreihen zusammen mit den zugehörenden ursprünglichen Zeitreihen zeigt Abbildung 8 in Abschnitt 6. Hierbei sind unterschiedliche Symbole durch Kreuze in verschiedenen Höhen dargestellt, wobei die Nullen weggelassen wurden. Die Symbole wurden nach dem Werteunterschied sortiert. Tief stehende Kreuze entsprechen stark oder leicht fallenden, mittlere Kreuze ungefähr waagerechten und hoch stehende Kreuze steigenden Mustern.

5 Suche nach Abhängigkeiten mit dem MSDD-Algorithmus

Zur Suche nach zeitlichen Abhängigkeitsbeziehungen zwischen Mustern wird der Multi-Stream Dependency Detection Algorithmus von Oates und Cohen eingesetzt [8]. Dieser leitet unter Vorgabe einer Anzahl k und einer Zeitverzögerung δ die k besten Abhängigkeiten der Form $x \overset{\delta}{\Rightarrow} y$ ab.

5.1 Der Raum möglicher Abhängigkeiten

Bei x und y handelt es sich um sogenannte „Multitoken“, die aus einfachen Token aufgebaut sind. Ein Token stellt ein Ereignis dar und ist als 3er Tupel $\tau = (s, v, d)$ definiert. Dabei gibt s einen Strom, d.h. die Nummer einer Zeitreihe S^s an, v ist ein Symbol dieser Zeitreihe und d gibt die Lage des Symbols relativ zum Beginn seines zugehörigen Multitokens an.

Betrachtet man als Beispiel die symbolischen Zeitreihen S^1, S^2 und S^3 der folgenden Tabelle und definiert x und y als Tupelmengen $x = \{(2, \mathsf{X}, 0), (3, 2, 0), (2, \mathsf{Y}, 1), (3, 1, 1)\}$ und $y = \{(1, \mathsf{A}, 0), (2, \mathsf{Z}, 0)\}$, so tritt das Multitoken x zu den Zeitpunkten 1 und 8 und das Multitoken y zu den Zeitpunkten 4 und 11 auf:

T:	1	2	3	4	5	6	7	8	9	10	11
S_1	D	B	B	A	D	C	D	B	C	A	A
S_2	X	Y	Y	Z	Y	X	Z	X	Y	X	Z
S_3	2	1	3	2	2	1	2	2	1	1	3

Der MSDD-Algorithmus ist in der Lage, sowohl besonders häufig vorkommende Multitoken als auch Zusammenhänge zwischen ihnen zu finden. Ein solcher Zusammenhang könnte bei Betrachtung einer Zeitverzögerung von $\delta = 3$ beispielsweise darin bestehen, daß nach x überdurchschnittlich häufig y folgt: $x \overset{3}{\Rightarrow} y$. Dabei wird x als „Precursor“ und y als „Successor“ bezeichnet.

Zur Ableitung wird eine systematische Suche im Raum aller Abhängigkeiten vorgenommen. Dieser wird neben der Menge der Ströme S angegeben durch die erlaubte maximale zeitliche Länge des Precursors w_p, die erlaubte maximale zeitliche Länge des Successors w_s und die Länge der Zeitverzögerung δ.

Bei m Strömen mit je n möglichen Symbolen beläuft sich die Anzahl der möglichen Precursor-Ereignisse auf $(n+1)^{mw_p}$. Ein Ereignis kann für einen Block von m Strömen auf einer Breite von w_p Werte angeben. An jeder dieser mw_p Stellen kann eines von n Token stehen oder der Wert dieses Tokens offengelassen werden. Entsprechend gibt es für den Successor $(n+1)^{mw_s}$ mögliche Multitoken. Die Abhängigkeitsbeziehungen $x \overset{\delta}{\Rightarrow} y$ geben 2 Multitoken (x und y) und einen Abstand (δ) an. Der Abstand ist ein konstanter Parameter des Algorithmus und somit für alle Abhängigkeiten gleich. Demnach ergibt sich die Gesamtzahl möglicher Abhängigkeiten als

$$(n+1)^{mw_p} * (n+1)^{mw_s} = (n+1)^{m(w_p+w_s)}.$$

Bestimmte Abhängigkeiten lassen sich unter Umständen durch verschiedene Multitoken ausdrücken, jedoch untersucht MSDD jeweils nur die Variante, bei der der Precursor zum frühestmöglichen Zeitpunkt beginnt.

5.2 Bewertung möglicher Abhängigkeiten

Abhängigkeiten können später als Wenn-Dann-Regeln dargestellt werden, wobei sowohl positive „Wenn x zum Zeitpunkt t, Dann y zum Zeitpunkt $t+\delta$“ als auch negative Zusammenhänge „Wenn x zum Zeitpunkt t, Dann $\overline{y}$ zum Zeitpunkt $t+\delta$“ zulässig sind. Zu ih-

rer Bewertung bei vorgegebener Zeitverzögerung wird zunächst eine Kontingenztabelle aufgestellt:

	y	$\overline{y}$	Σ
x	n_1	n_2	r_1
$\overline{x}$	n_3	n_4	r_2
Σ	c_1	c_2	T

n_1 ist hierbei die Anzahl von Zeitpunkten der Zeitreihen, zu denen zunächst x und δ Zeitschritte später y auftrat, n_2 ist die Anzahl Zeitschritte, bei denen nach x nicht y auftrat usw.. r_1 gibt an, wie oft x auftrat, r_2, wie oft x nicht auftrat, c_1, wie of y auftrat und c_2, wie oft y nicht auftrat. r_1 und r_2 bzw. c_1 und c_2 addieren sich zur Gesamtanzahl an Beobachtungszeitpunkten T.

Anschließend werden aus den Randverteilungen unter der Hypothese der statistischen Unabhängigkeit Erwartungswerte für die n_i gebildet, z.B.

$$\hat{n}_1 = \hat{p}(x \wedge y)T = \hat{p}(x)\hat{p}(y)T = \frac{r_1}{T}\frac{c_1}{T}T = \frac{r_1 c_1}{T}$$

Die Bewertung einer Regel liefert nun die G-Statistik [11], die auch als Likelihood-Ratio-Statistik bezeichnet wird und mit einem χ^2-Test verglichen werden kann:

$$G = 2\sum_{i=1}^{4} n_i \log \frac{n_i}{\hat{n}_i}$$

Angesichts der hohen Anzahl möglicher Regeln wäre es unmöglich, den Hypothesenraum vollständig zu durchsuchen. Die Anwendbarkeit des MSDD-Algorithmus beruht darauf, die Güte von Regeln, die von einer Regel durch Spezialisierung, d.h. durch Hinzufügen von Token zu Precursor oder Successor, abgeleitet werden, abschätzen zu können.

Handelt es sich um eine Regel mit nicht-leerem Successor und liegen die Werte $n_1, \ldots, n_4$ der Kontingenztabelle vor, so kann jede davon abgeleitete Regel höchstens eine Güte von:

$$Gmax_1(n_1, n_2, n_3, n_4)$$
$$= \max \begin{pmatrix} G(n_1, n_2, 0, n_3 + n_4) \\ G(0, n_1 + n_2, n_3, n_4) \end{pmatrix}$$

erreichen.

Für eine Regel mit leerem Successor erhält man als Abschätzung:

$$Gmax_2(n_1, n_2, n_3, n_4)$$
$$= \max \begin{pmatrix} (1) \\ \quad G(n_1, 0, 0, n_2 + n_3 + n_4), \\ \qquad \text{falls } n_1 \leq n_2 + n_3 + n_4 \\ \quad G(\frac{n_1+n_2+n_3+n_4}{2}, 0, 0, \frac{n_1+n_2+n_3+n_4}{2}), \\ \qquad \text{sonst} \\ (2) \\ \quad G(0, \frac{n_1+n_2+n_3}{2}, \frac{n_1+n_2+n_3}{2}, n_4), \\ \qquad \text{falls } n_1 \leq abs(n_2 - n_3) \\ \quad G(0, n_2, n_1 + n_3, n_4), \\ \qquad \text{falls } n_2 > n_3 \text{ und } n_1 > abs(n_2 - n_3) \\ \quad G(0, n_1 + n_2, n_3, n_4), \\ \qquad \text{sonst} \end{pmatrix}.$$

5.3 Der MSDD-Algorithmus

MSDD($\mathcal{S}, w_p, w_s, \delta, f, k$)

1. *best* = ()
2. *open* = ({} ⇒ {})
3. while (not EMPTY(*open*)) do
   (a) *node* = NEXT-NODE(*open*)
   (b) *children* = SYSTEMATIC-EXPAND(*node*, w_p, w_s)
   (c) add *children* to *open*
   (d) for *child* in *children* do
      i. if (LENGTH(*best*) $< k$ or $\exists n \in$ *best* : f(*child*, $\mathcal{S}$, δ) > $f(n, \mathcal{S}, \delta)$) then add *child* to *best*
      ii. if (LENGTH(*best*) $> k$) then remove from *best* the node with the lowest f value
4. return *best*

Abbildung 7 Pseudocodedarstellung des MSDD-Algorithmus

Die Pseudocode-Darstellung von Abbildung 7 gibt einen Überblick über den MSDD-Algorithmus.

Zum Finden der k besten Abhängigkeiten, die in $best$ gesammelt werden, wird eine Liste möglicher Abhängigkeiten $open$ verwaltet. Ausgehend von der „leeren“ Abhängigkeit $\{\} \stackrel{\delta}{\Rightarrow} \{\}$ wird innerhalb einer Schleife jeweils eine Abhängigkeit $node$ erweiter (spezialisiert) (SYSTEMATIC-EXPAND). Die Erweiterungs-Methode stellt dabei durch Definition einer Ordnung auf allen möglichen Abhängigkeiten sicher, daß jeder Kandidat nur höchstens einmal generiert wird (optimal refinement operator). Danach werden solche neuen Abhängigkeiten ($children$), die durch weitere Spezialisierungen sicher nicht mehr zum Ergebnis beitragen können, entfernt (REMOVE-PRUNABLE). Hierzu werden die oben beschriebenen Abschätzungen für G verwendet, und in der dadurch erreichten effizienten Beschneidung des Suchbaumes liegt die Stärke des Algorithmus. Anschließend werden die verbleibenden Kandidaten ($children$) zur Liste $open$ hinzugefügt, mit Hilfe der Bewertungsfunktion f bewertet und ggf. zur Bestenliste hinzugefügt. Die Schleife wird solange wiederholt, bis alle $node$s der $open$-Liste abgearbeitet sind.

5.4 Erweiterung der Zeitsemantik

Der MSDD erkennt nur Zusammenhänge zwischen zeitlich exakt im Abstand δ auftretenden Ereignissen. Dies führt bei Anwendung auf aus numerischen Meßwertreihen generierte symbolische Zeitreihen (siehe Abschnitt 4) zu schlechten Ergebnissen, da dort zeitliche Unschärfen auftreten. Diese können sowohl in der Natur des analysierten dynamischen Systems liegen als auch in Störungen oder Vorverarbeitungsschritten begründet sein und führen häufig dazu, daß entweder ein Zusammenhang nicht erkannt wird oder aber meh-

rere Regeln gefunden werden, die alle denselben Zusammenhang beschreiben.

Zeitliche Unschärfen sollten jedoch nicht nur zwischen Precursor und Successor, sondern auch innerhalb der Multitoken berücksichtigt werden. Aus diesem Grund wurde zwar die Vorgabe einer festen, scharfen Zeitverzögerung δ beibehalten, jedoch wurde die Zeitsemantik der einzelnen Token $\tau = (s, v, d)$ erweitert. Hierzu wurde der Offset d zu einem Intervall $[d_1; d_2]$ erweitert, und ein solches Token $\tau'(s, v, [d_1; d_2])$ dann als gültig angesehen, wenn das Symbol v im Strom S^s innerhalb des Zeitintervalls $[d_1; d_2]$ relativ zum Beginn des zugehörenden Multitokens auftritt.

Um eine effiziente Berechnung zu ermöglichen, wurde eine Menge möglicher Zeitintervalle als zusätzlicher Parameter eingeführt. Außerdem wurden die Parameter w_p und w_s dahingehend uminterpretiert, daß sie nicht mehr die zeitliche Ausdehnung der Multitoken begrenzen, sondern die maximale Anzahl von Token vorgeben.

Im Ergebnis haben die entstehenden Regeln $x \overset{\delta}{\Rightarrow} y$ beispielsweise bei positiven Zusammenhängen wieder die Form „Wenn x zum Zeitpunkt t, Dann y zum Zeitpunkt $y + \delta$“, wobei beachtet werden muß, daß x und y den Großteil der Zeitsemantik beinhalten und δ nur dazu dient, den Precursor zeitlich vor dem Successor zu fixieren. Würde man δ zu allen Zeitintervallgrenzen hinzuaddieren, könnte man auf die Nennung verzichten.

6 Anwendungsbeispiel

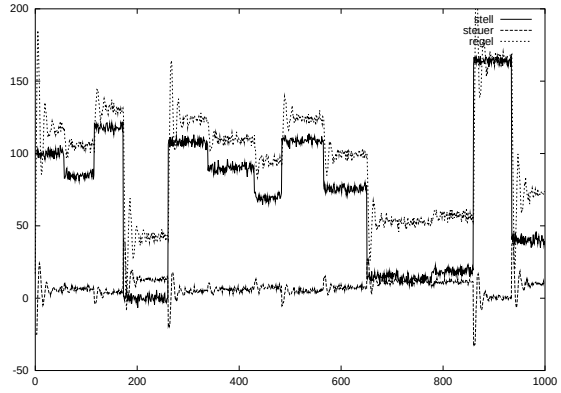
Ein erster Test des Verfahrens wurde an Simulationsdaten eines einfachen dynamischen Systems mit Totzeit vorgenommen. Bei dem aus der kognitionspsychologisch orientierten Literatur stammende Kühlhausexperiment [10] läßt sich über ein Stellrad die Temperatur eines Kühlhauses beeinflussen und mittels eines Thermometers die Innentemperatur beobachten. Die Innentemperatur $regel$ hängt dabei nur indirekt über eine Steuergröße $steuer$ von der Stellgröße $stell$ ab:

$$\begin{aligned} steuer_i &= (regel_{i-v} - stell_i) * regelfaktor \\ regel_i &= regel_{i-1} - steuer_{i-1} \\ &\quad + (stoer_i - regel_{i-1}) * tempo \end{aligned}$$

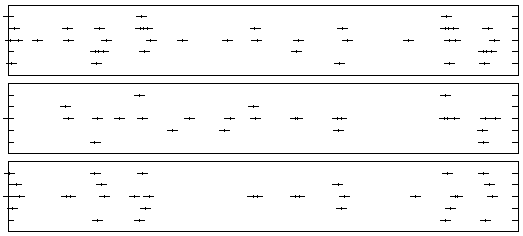
Zur Simulation wurden die Stellwerte durch einen Zufallsgenerator vorgegeben, der in zufälligen Abständen die Stellgröße $stell_i$ veränderte. Die resultierenden drei Zeitreihen wurden mit einer normalverteilten Störung überlagert.

Abbildung 8 zeigt einen Ausschnitt aus einer Simulation mit folgenden Parametern:

$$\begin{aligned} stoer_i &= 170, \\ tempo &= 0,1 \\ v &= 3 \\ regelfaktor &= 0,3 \end{aligned}$$



(a) Original-Zeitreihen



(b) Symbolische Zeitreihen für Regel-, Stell- und Steuergröße (von oben nach unten)

Abbildung 8 Simulationsergebnisse „Kühlhausexperiment“

Dabei sind im oberen Teil die Simulationsdaten und darunter die generierten symbolischen Zeitreihen dargestellt (Erläuterung siehe Abschnitt 4.3).

Nr.	Regel	$(n_1; \ldots; n_4)$	G
1.	(Steuer,↘,[-3;-1]) ⇒ (Regel,↗,[0;2])	(181;92;175;9552)	972
2.	(Regel,↓,[-6;-4]) ⇒ (Steuer,↗,[3;5])	(116;25;199;9660)	719
3.	(Regel,↗,[-3;-1]) ⇒ (Steuer,↗,[0;2])	(150;206;165;9479)	644
4.	(Regel,↓,[-6;-4]) ⇒ (Regel,↗,[0;2])(Steuer,↗,[3;5])	(89;52;61;9798)	630
5.	(Steuer,↗,[-3;-1]) ⇒ (Regel,↘,[0;2])	(127;188;125;9560)	592
6.	(Regel,↓,[-6;-4]) ⇒ (Regel,↗,[0;2])	(106;35;250;9609)	585
7.	(Regel,↓,[-6;-4])(Regel,↗,[0;2]) ⇒ (Steuer,↗,[3;5])	(89;17;226;9668)	550
8.	(Steuer,↘,[-3;-1]) ⇒ (Regel,↗,[0;2])(Steuer,↗,[3;5])	(96;177;54;9673)	535
9.	(Stell,↑,[0;2]) ⇒ (Regel,↗,[0;2])(Steuer,↓,[0;2])	(46;14;0;9940)	522
10.	(Regel,↗,[0;2])(Steuer,↓,[0;2]) ⇒ (Stell,↑,[0;2])	(46;0;14;9940)	522

Abbildung 9 Gefundene Regeln des Kühlhausexperimentes

Zur Regelsuche mit dem erweiterten MSDD-Algorithmus wurden die Zeitintervalle $[-6; -4]$, $[-3; -1]$ und $[0; 2]$ für den Precursor sowie $[0; 2]$,

$[3;5]$ und $[5;7]$ für den Successor verwendet. Die Zeitverzögerung δ ist hierbei schon in den Intervallen eingerechnet, und sie sind so verschoben, daß die Nullstelle bei der Überschneidung von Precursor und Successor liegt. Die zehn besten gefundenen Regeln sind in Abbildung 9 tabellarisch zusammengefaßt. Der MSDD benötigte zur Berechnung der Regeln in Java auf einem Pentium III 450 Mhz zwei Minuten. In dieser Zeit wurden 539 Knoten expandiert und 22200 Abkömmlinge verworfen.

Betrachtet man die Werte von n_1 und n_2, so kann die erste Regel als positiver Zusammenhang interpretiert werden: „Wenn $steuer$ im Zeitintervall $[-3;-1]$ mittelstark abfiel, dann wird $regel$ mit einer Wahrscheinlichkeit von $66,3\%$ im Zeitintervall $[0;2]$ mittelstark ansteigen." Demgegenüber weist Regel 3 auf einen negativen Zusammenhang zwischen $regel$ und $steuer$ hin. Beide Regeln zusammen lassen sich als Hinweis auf gegenläufiges Verhalten beider Größen mit einer Verzögerung von etwa 3 Zeitschritten interpretieren, was auch den Eigenschaften des simulierten Kühlhauses entspricht.

Weiterhin erkennt man, daß die Regeln zum Teil redundant sind. So handelt es sich beispielsweise bei Regel 4 um eine Zusammenfassung der Regeln 2 und 6. Problematische Auswirkungen der zeitlichen Unschärfe zeigen die Regeln 9 und 10. Beide beschreiben denselben Zusammenhang, können aber nicht zwischen Ursache und Wirkung unterscheiden.

7 Zusammenfassung und Ausblick

Neben der im vorhergehenden Abschnitt gezeigten Anwendung auf Simulationsdaten konnten in [7] bereits erste positive Ergebnisse mit Daten einer energetischen Verwertungsanlage erzielt werden. Dies zeigt die prinzipelle Eignung des Verfahrens zur Analyse des Verhaltens realer dynamischer Systeme. Es existiert jedoch eine ganze Reihe von Verbesserungsmöglichkeiten.

Am Verfahren selbst ist insbesondere die manuelle Vorgabe der Zeitintervalle unbefriedigend. Daher wird an der Schätzung geeigneter Zeitverzögerungen, beispielsweise mittels zeitlich versetzter (Auto-) Korrelationsmatrizen, gearbeitet.

Bei der Analyse von Ergebnissen fiel auf, daß viele gefundene Regeln durch eine Nachbearbeitung verbessert werden könnten. Durch Erweiterung von Zeitintervallen könnten Regeln zusammengefaßt bzw. durch Verkleinerung spezialisiert werden.

Die Anwendbarkeit des MSDD-Algorithmus auf komplexere Systeme, die beispielsweise eine hohe Anzahl von Zeitreihen aufweisen, wird durch die Berechnungskomplexität eingeschränkt, die durch die Erweiterung der Zeitsemantik nochmals erhöht wurde. Während einfache Systeme, wie z.B. das in Abschnitt 6 gezeigte, innerhalb von wenigen Minuten analysiert werden können, muß bei größeren Problemen mit Laufzeiten im Stunden- oder Tagesbereich gerechnet werden. Eine Verbesserung ließe sich beispielsweise durch Parallelisierung erreichen [9].

Längerfristige Forschungsaktivitäten betreffen die Erweiterung des Verfahrens von einem Analyse- zu einem Modellierungswerkzeug. Hierzu sind Methoden zu entwickeln, um aus den gefundenen statistischen Zusammenhängen zunächst anwendbare Einzelregeln und danach konsistente Regelsätze zu generieren.

References

1. James C. Bezdek. *Fuzzy Mathematics in Pattern Classification*. PhD thesis, Center for Applied Mathmatics, Cornell University, New York, 1973.
2. James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.
3. Michael Boronowsky. Effiziente Interpretation multipler Meßdaten mittels Entscheidungsbauminduktion. In Andreas Fick and Hubert B. Keller (Editoren), *Workshop Wissensbasierte/Intelligente Systeme in Umweltanwendungen*, pages 39–47. Forschungszentrum Karlsruhe GmbH, 1999.
4. J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact, well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1974.
5. Rolf Isermann. *Identifikation dynamischer Systeme. Band 1*. Springer-Verlag, Berlin Heidelberg, 1988.
6. Jens Jäkel, Ralf Mikut, Hagen Malberg, and Georg Bretthauer. Datenbasierte Regelsuche für Fuzzy-Systeme mittels baumorientierter Verfahren. In *9. Workshop Fuzzy Control des GMA-UA 1.4.2*, 1999.
7. Markus Kraft. *Ableitung von Ursache-Wirkungs-Beziehungen über Musteranalyse von Zeitreihen*. Diplomarbeit, Universität Karlsruhe, 2001.
8. Tim Oates and Raul R. Cohen. Searching for structure in multiple streams of data. In *Proceedings of the Thirteens International Conference on Machine Learning*, pages 346–354, 1996.
9. Tim Oates, Matthew D. Schmill, and Paul R. Cohen. Efficient mining of statistical dependencies. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 794–799, 1999.
10. Ute Reichert and Dietrich Dörner. Heurismen beim Umgang mit einem "einfachen" dynamischen System. *Sprache & Kognition*, 7(1):12–24, 1988.
11. Thomas D. Wickens. *Multiway Contingency Tables Analysis for the Social Sciences*. Lawrence Erlbaum Associates, 1989.
12. Hans-Jürgen Zimmermann (Editor). *Proceedings / EUFIT '99 : 7th European Congress on Intelligent Techniques & Soft Computing ; Aachen, Germany, September 13-16, 1999*, Aachen , Mainz, 1999.