

Walter Frisch
Alfred Taubes (Hrsg.)

Informations- Wirtschaft

Aktuelle Entwicklungen und Perspektiven
Symposium, Wien, 29./30. September 1993

Mit 87 Abbildungen

Physica-Verlag

Ein Unternehmen
des Springer-Verlags

Zur Beschleunigung des Lernens genetischer Algorithmen mittels unscharfer Regelsprachen

Andreas Geyer-Schulz¹

Institut für Betriebswirtschaftslehre – Schwerpunkt Wirtschaftsinformatik
Wirtschafts- und Sozialwissenschaftliche Fakultät
Universität Augsburg, Memmingerstraße 18
D-8900 Augsburg, Deutschland

Zusammenfassung: Unscharfe Classifiersysteme sind evolutionsgesteuerte Maschinlernsysteme, die eine unscharfe Regelbasis, einen genetischen Algorithmus und eine Bewertungsfunktion integrieren. In diesem Beitrag wird gezeigt, daß die Rechenkomplexität eines Classifiersystems durch die syntaktische Struktur der vom Produktionsregelinterpret verwendeteten formalen Sprache L bestimmt wird. Von besonderem Interesse ist dabei, die Rechenkomplexität eines unscharfen Classifiersystems mit der entsprechenden Version eines klassischen Classifiersystems zu vergleichen. Mittels eines solchen Vergleichs kann dann gezeigt werden, daß unscharfe Classifiersysteme theoretisch einen unendlichen Speed-Up im Vergleich zu klassischen Classifiersystemen liefern. Sogar praktische Rechnerimplementierungen unscharfer Classifiersysteme, die ja unscharfe Classifiersysteme nur approximieren, lernen beträchtlich schneller als klassische Classifiersysteme.

I. Einführung

In seinem 1936 in "Beiträge zu einem mathematischen Kolloquium" erschienenen Artikel "Über die Länge von Beweisen" hat Kurt Gödel erstmals das Problem behandelt, die Länge von Beweisen von in zwei verschiedenen formalen Systemen beweisbaren Formeln zu vergleichen. Gödels ohne Beweis präsentiertes Speed-Up Theorem besagt, daß der Übergang von einem formalen System zu einem formalen System höherer Ordnung die Länge von Beweisen verkürzen kann [Gödel, 1936; Feferman *et al.*, 1986]. Analog dazu hat Parikh [Parikh, 1973] ein Speed-Up Theorem für ein System der Peano Arithmetik mit vollständiger Induktion und mit Addition und Multiplikation als ternären Relationen als System niedriger Ordnung und Analysis als System höherer Ordnung bewiesen. [Statman, 1978] enthält eine Studie des Speed-Up Phänomens für verschiedene Kombinationen klassischer Systeme.

Der Begriff *Speed-Up* stammt von Blum, der 1967 das berühmte Speed-Up Theorem der Theorie rekursiver Funktionen entdeckte [Blum, 1967]. Blums

¹Beurlaubt: Abteilung für Angewandte Informatik insbesondere Betriebsinformatik, Institut für Informationsverarbeitung und Informationswirtschaft, Wirtschaftsuniversität Wien, Augasse 2-6, A-1090 Wien. Telefon: +43-1-3303636-31. EMAIL: geyers at wu-wien.ac.at

- [57] White, H., Learning in Artificial Neural Networks: A Statistical Perspective. Neural Computation, 1, 425-464 (1989a)
- [58] White, H., Some Asymptotic Results of Learning in Single Hidden-Layer Feedforward Network Models, Journal of the American Statistical Association, Vol. 84, No. 408, 1003-1013 (1989b)
- [59] Whitley, D., Dominic, S. and Das R., Genetic Reinforcement Learning with Multilayer Neural Networks, in: Proc. Fourth Intern. Conf. Genetic Algorithms, Morgan Kaufmann, San Mateo, CA (1991)
- [60] Zimmermann, H.G., Hergert, F., Finnoff, W. Neuron Pruning and Merging Methods for Use in Conjunction with Weight Elimination, Siemens AG. Corporate Research and Development, ZFE IS INF 23, Otto-Hahn-Ring 6, 8000 München 83 (1992)

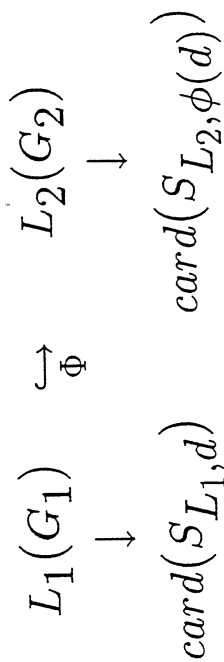


Abbildung 1: Der Vergleich von zwei Suchräumen

formalen Potenzreihen [Goulden and Jackson, 1983; Kuich and Salomaa, 1986; Salomaa, 1990; van Leeuwen, 1990] ausgenutzt wird, kann darüber hinaus die Definition der formalen Potenzreihen $A(x)$ aus den Produktionsregeln P einer Grammatik G der Sprache L automatisch durch eine Signaturabbildung in der Form einer Menge von Rekurrenzrelationen gewonnen werden. In Abbildung 1 wird eine solche Signaturabbildung zweimal verwendet, nämlich um die Suchraumgrößen $\text{card}(S_{L_1,d})$ aus $L_1(G_1)$ und $\text{card}(S_{L_2,d})$ aus $L_2(G_2)$ abzuleiten. Die Pfeile $L_1(G_1) \rightarrow \text{card}(S_{L_1,d})$ und $L_2(G_2) \rightarrow \text{card}(S_{L_2,\phi(d)})$ symbolisieren diese Signaturabbildungen in Abbildung 1.

Die Grundidee, die den Vergleich der beiden so abgeleiteten Reihen gestattet, besteht nun darin, unter Ausnutzung der Übersetzung $\Phi : L_1(G_1) \hookrightarrow L_2(G_2)$ für jedes Glied der ersten Reihe das entsprechende Glied der zweiten Reihe, mit dem es verglichen werden soll, zu bestimmen. Aus der Übersetzung $\Phi : L_1(G_1) \hookrightarrow L_2(G_2)$ wird dabei zunächst die Funktion $\phi : N \rightarrow N_0$ konstruiert, die die Beschränkung in der Ableitungstiefe d der Grammatik G_1 auf eine Beschränkung der Ableitungstiefe $\phi(d)$ für die Grammatik G_2 abbildet. Diese *Ableitungstiefenbeschränkungsfunktion* $\phi(d)$ stellt sicher, daß für jede Ableitungstiefe d von $L_1(G_1)$ $\phi(d)$ die kleinste Ableitungstiefe in $L_2(G_2)$ ist, sodaß alle Übersetzungen $\Phi(w)$ aller Worte w von $S_{L_1,d}$ in $S_{L_2,\phi(d)}$ enthalten sind. Φ wird auch als *Beschränkungsfunktion* bezeichnet, da Φ die Ableitungstiefenbeschränkungsfunktion $\phi(d)$ induziert, die für jeden Koeffizienten der ersten Reihe den Index des Koeffizienten der zweiten Reihe angibt, mit dem er verglichen werden soll. Abbildung 1 stellt diese Idee im Überblick dar.

Alles, was jetzt noch übrigbleibt, ist sicherzustellen, daß fast überall

$$\text{card}(S_{L_1,d}) < \text{card}(S_{L_2,\phi(d)})$$

gilt. Das kann sich jedoch als ziemlich schwierige Aufgabe erweisen. Glücklicherweise genügt es für die meisten praktischen Anwendungen, die Koeffizienten der beiden formalen Potenzreihen bis zu einem fixen (und kleinen) n zu berechnen, da in Anwendungen (zumindest für unser Beispiel) die größte Ableitungstiefe, die verwendet wird, ziemlich klein ist.

Speed-Up Theorem beweist die Existenz einer vollständig berechenbaren Funktion, für die kein bestes Programm existiert [Young, 1973; Cutland, 1980]. Salomaa bewies ein Speed-Up Theorem für zeitbeschränkte Grammatiken [Salomaa, 1973, p. 304]. Salomaas Speed-Up Theorem liegt folgende Idee zu Grunde. Zu einer Grammatik G_1 wird eine gleichwertige Grammatik G_2 konstruiert, sodaß m Ableitungsschritte in G_1 in einem einzigen Schritt in G_2 durchgeführt werden können. Zum Beispiel, $G_1 = \{S \rightarrow aS \mid a\}$ wird zu $G_2 = \{S \rightarrow aS \mid a \mid aa\}$ transformiert.

Die Motivation, das Speed-Up Phänomen näher zu untersuchen, liegt in der folgenden Tatsache. Offensichtlich kann durch die Wahl einer geeigneten sehr hohen Produktionsregelsprache eine beträchtliche Verringerung der Lernkomplexität eines Classifiersystems erzielt werden. Es scheint zudem, daß für Aufgaben wie formale Programmverifikation, automatisches Programmtesten, automatisches Programmieren und automatisches Beweisen, deren Rechenaufwand und, da der Rechenaufwand den Engpaßfaktor darstellt, deren Durchführbarkeit hauptsächlich von der Suchraumgröße beeinflusst wird, Abstraktion im Sinne eines Übergangs zu formalen Systemen höherer Ordnung den einzigen Lösungsweg darstellt. Im folgenden wird nun gezeigt werden, daß unscharfe Produktionsregelsprachen im Vergleich zu klassischen Produktionsregelsprachen solche höheren formalen Systeme darstellen.

II. Der Vergleich von zwei Suchräumen

Wie kann ganz allgemein die Lernkomplexität zweier kontextfreier Sprachen, die durch eine Übersetzung von der einen Sprache in die andere miteinander verbunden sind, verglichen werden?

Um diese Frage beantworten zu können, wird im folgenden, um etwaige Zählproubleme zu vermeiden, angenommen, daß

1. die Grammatiken G_1 und G_2 der Sprachen L_1 und L_2 ϵ -frei sind,
2. keine nicht benötigten Symbole enthalten sind, und
3. in der gleichen Normalform [Aho and Ullman, 1972] sind oder, wie in unserem in Abschnitt V. gezeigten Beispiel, fast gleiche Struktur (Symbol-Symbol Abbildungen) haben.

Die (inhärente) Mehrdeutigkeit der Grammatik wird beseitigt, indem ein Wort w aus $L(G)$ immer wie folgt definiert wird: Ein Wort w aus $L(G)$ besteht aus einem Paar. Das erste Element des Paares ist die Kette der Terminalsymbole (wie gewohnt), das zweite Element ist der Linksableitungsbaum des ersten Elements.

Betrachtet man nun die in der Anzahl der Ableitungen beschränkte Version von $L(G)$, so kann man die Lernkomplexität durch eine formale Potenzreihe $A(x) = \sum_{d \geq 0} \text{card}(S_{L,d}) \cdot x^d$ charakterisieren, deren Koeffizienten $\text{card}(S_{L,d})$ der Anzahl der Worte, die mit höchstens d Ableitungen gebildet werden können, entspricht. Indem eine natürliche Verbindung zwischen formalen Sprachen und

(a) Für alle Elemente von V_T ist F_1 definiert durch

$$\forall x \in V_T : x \rightarrow \Pi(x, d - 1).$$

Wir fügen für alle $x \in V_T$ eine Klausel der folgenden Art zum rekursiven Wortzählfunktionsschema Π hinzu:

$$\Pi(x, d) = 1 \cdot (d = 0)$$

Für ein Terminalsymbol x ist $\Pi(x, d)$ 1, wenn $d = 0$ und 0 in allen anderen Fällen.

(b) Für alle Elemente von V_N ist F_1 definiert durch

$$\forall y_{LHS} \in V_N : y_{LHS} \rightarrow \Pi(y_{LHS}, d)$$

$$\forall y_{RHS} \in V_N : y_{RHS} \rightarrow \Pi(y_{RHS}, r_i)$$

Mit einem Nichtterminalsymbol y und $d < 1$ als Argumente ist Π nicht definiert. Immer wenn Π nicht definiert ist, erhält Π den Wert 0, um eine vollständig definierte Funktion zu erhalten. Für r_i , siehe Ausdruck 3.

(c)

$$:= \rightarrow =$$

(d)

$$| \rightarrow +$$

(e)

$$* \rightarrow o$$

In der BNF-Notation wird die Verkettung zweier Symbole $x_1 * x_2$ üblicherweise durch unmittelbares Nebeneinanderstellen der beiden Symbole dargestellt: $x_1 x_2$. Jede k -Symbolkette wird deshalb durch folgenden Ausdruck ersetzt:

$$\sum_{\sum_{i=1}^k r_i = d-1, r_i \geq 0, d > 0} \prod_{i=1}^k \Pi(y_i, r_i) \quad (3)$$

Ausdruck 3 scheint geradewegs vom Himmel zu fallen. Seine Ableitung kann jedoch wie folgt erklärt werden: Man betrachte zuerst den Fall einer Symbolkette aus zwei Symbolen, $x_1 x_2$ und es seien e Ableitungsschritte noch verfügbar. Wieviele verschiedene Worte können in e Ableitungsschritten aus der Symbolkette $x_1 x_2$ generiert werden? Die Antwort erhält man aus der Analyse der folgenden vier Fälle:

III. Ein Meta-Theorem

Für zwei konkrete, kontextfreie Sprachen ist dann jeweils folgendes Theorem zu beweisen.

Theorem 1 Gegeben sind zwei kontextfreie Sprachen, $L_1(G_1)$ und $L_2(G_2)$, und eine effektive (berechenbare) Einbettung $\Phi : L_1(G_1) \hookrightarrow L_2(G_2)$ (eine Übersetzung von $L_1(G_1)$ nach $L_2(G_2)$), dann gilt $\text{card}(S_{L_1, d}) < \text{card}(S_{L_2, \Phi(d)})$, für fast alle $d \in \mathbb{N}$.

Die Beweisskizze folgt im nächsten Abschnitt.

IV. Eine Beweisskizze

- Die Pfeile, $L_1(G_1) \rightarrow \text{card}(S_{L_1, d})$ beziehungsweise $L_2(G_2) \rightarrow \text{card}(S_{L_2, \Phi(d)})$, bezeichnen rekursive Wortzählfunktionen (WCF), die die Anzahl der Worte in $S_{L_1, d}$ und $S_{L_2, \Phi(d)}$ zählen. Für $L(G)$ kann die rekursive Wortzählfunktion Π automatisch mittels der Signaturabbildung $F_1 : \sum_{BNF} \rightarrow \sum_{WCF}$ aus den Produktionsregeln P von $L(G)$, die üblicherweise in Backus-Naur Form (BNF) angegeben werden, abgeleitet werden.

$$\sum_{BNF} = \langle V_T, V_N, :=, |, * \rangle \quad (1)$$

ist die Signatur der BNF-Sprache, wobei V_T das Alphabet der Terminalsymbole bezeichnet, V_N das Alphabet der Nichtterminalsymbole bezeichnet, $:=$ wird abgeleitet zu bezeichnet, $|$ oder bezeichnet, und $*$ die Verkettung von Symbolen bezeichnet. (Üblicherweise wird die Verkettung einfach durch Nebeneinanderstellen von Symbolen dargestellt.) Da $y \in V_N$ auf beiden Seiten einer Produktionsregel vorkommen kann, bezeichnet y_{LHS} das Vorkommen von y auf der linken Seite und y_{RHS} das Vorkommen auf der rechten Seite einer Produktionsregel.

Die Signatur der rekursiven Wortzählfunktion ist

$$\sum_{WCF} = \langle \Pi(x, d), \Pi(y, d), =, +, o \rangle \quad (2)$$

$\Pi(x, d)$ bezeichnet die Menge aller Aufrufe und Definitionen der rekursiven Wortzählfunktion Π mit $x \in V_T$ und $d \in \mathbb{N}_0$ als Argumente. $\Pi(y, d)$ bezeichnet die Menge aller Aufrufe und Definitionen der rekursiven Wortzählfunktion Π mit $y \in V_N$ und $d \in \mathbb{N}$ als Argumente. = bezeichnet ist definiert durch und + bezeichnet die Addition. o ist eine Art ternäre Funktionskompositionsoption. $o(\Pi(y_1, d), \Pi(y_2, d), d)$ berechnet die Anzahl der Worte, die von der Symbolkette $y_1 y_2$ in d Ableitungen erzeugt werden können. Das rekursive Wortzählfunktionsschema $\Pi : (V_T \times \mathbb{N}_0) \cup (V_N \times \mathbb{N}) \rightarrow \mathbb{N}_0$ wird nun durch die Signaturabbildung $F_1 : \sum_{BNF} \rightarrow \sum_{WCF}$ abgeleitet:

- i. Beide Symbole x_1 und x_2 sind Nichtterminalsymbole. Um nun eine Kette von Terminalsymbolen abzuleiten, muß man x_1 zum Beispiel r_1 Ableitungen zuordnen. Für diese Aufteilung von e in zwei Teile (2-Partition) ist die Anzahl der ableitbaren Worte das Produkt der Anzahl der von x_1 in r_1 Ableitungen erzeugbaren Worte mit der Anzahl der von x_2 in r_2 Ableitungen erzeugbaren Worte. Um alle Worte zu finden, muß man über alle 2-Partitionen von e summieren. Das bedeutet, alle Partitionen, die durch Paare r_1, r_2 unter der Bedingung, daß sowohl $r_1 + r_2 = e$ und $r_1, r_2 \neq r_2, r_1$ gilt, erzeugt werden können:

$$o(\Pi(x_1, e), \Pi(x_2, e), e) = \sum_{r_1+r_2=e, r_1 \geq 0, r_2 \geq 0, e \geq 0} \Pi(x_1, r_1) \cdot \Pi(x_2, r_2) \quad (4)$$

- Wenn $r_i = 0$ gilt, ist $\Pi(x_i, r_i)$ nicht definiert, da keine Kette von Terminalsymbolen abgeleitet werden kann und dies impliziert, daß $\Pi(x_i, r_i) = 0$ gilt.
- ii. x_1 ist ein Terminalsymbol und x_2 ein Nichtterminalsymbol. Ausdruck 4 liefert auch in diesem Fall das Ergebnis. Der Leser überzeuge sich, daß dies der Fall ist, weil $\Pi(x_1, r_1)$ für alle Werte von r_1 außer für $r_1 = 0$ den Wert 0 ergibt.
- iii. x_1 ist ein Nichtterminalsymbol und x_2 ein Terminalsymbol. Dies entspricht dem vorigen Fall.
- iv. Beide Symbole x_1 und x_2 sind Terminalsymbole. Ausdruck 4 ist nur definiert, wenn e, r_1 und r_2 den Wert 0 haben. In diesem Fall ist das Ergebnis 1.

Um Ausdruck 3 zu erhalten, muß man Ausdruck 4 auf den Fall einer n -Partition unter Berücksichtigung der Terminalsymbole verallgemeinern. Dies führt offensichtlich zu einer Verallgemeinerung der $2 + 1$ -ären Operation o zu einer $n + 1$ -ären Operation.

Schließlich kann nun die Suchraumgröße $card(S_{L,d})$ von L für alle $d \in N$ mittels der Potenzreihe $A(x)$ dargestellt werden, deren Koeffizienten wie folgt definiert sind:

$$card(S_{L,d}) = \sum_{i=1}^d \Pi((startsymbol), i).$$

- Diese Reihe ist monoton steigend mit Koeffizienten in N_0 .
2. Die Ableitungsbeschränkungsfunktion wird so definiert, daß das Bild $\Phi(w)$ von jedem Wort w , das mit höchstens d Ableitungen erzeugt werden kann, mit höchstens $\phi(d)$ Ableitungen in L_2 erzeugt werden kann und daß $\phi(d)$ die kleinste natürliche Zahl ist, für die das gilt. Zur Vereinfachung wird die Konstruktion von $\phi(d)$ in zwei Schritte zerlegt. Im ersten Schritt wird dabei eine rekursive Ableitungstiefenzählfunktion aus L_1 abgeleitet, im

zweiten Schritt wird diese Funktion so modifiziert, daß die Übersetzung Φ berücksichtigt wird. Im Einzelnen ist dabei wie folgt vorzugehen:

- (a) Die partielle Ableitungstiefenzählfunktion $\varphi : (V_T \times N_0) \cup (V_N \times N) \rightarrow N_0$ wird aus den Produktionsregeln von G_1 von L_1 durch die Signaturabbildung $F_2 : \Sigma_{BNF} \rightarrow \Sigma_{DCF}$ gewonnen. Durch die Hinzunahme zusätzlicher Regeln für den Wert *nicht definiert* wird die partielle Funktion vervollständigt, um leicht auf einem Rechner implementiert werden zu können. Offensichtlich ist auch diese Abbildung vollständig allgemein.

Die Signatur der rekursiven Ableitungstiefenzählfunktionsprache ist

$$\sum_{DCF} = (\varphi(x, d), \varphi(y, d), =, s(w), \max, \bullet) \quad (5)$$

$\varphi(x, d)$ ist die Menge aller Aufrufe und Definitionen der rekursiven Ableitungstiefenzählfunktion φ mit $x \in V_T$ und $d \in N_0$ als Argumente. $\varphi(y, d)$ ist die Menge aller Aufrufe und Definitionen der rekursiven Ableitungstiefenzählfunktion φ mit $y \in V_N$ und $d \in N$ als Argumente. $=$ $s(w)$ bezeichnet *ist definiert durch 1 plus "dem Rest"*. \max bezeichnet die größere von zwei Zahlen und \bullet ist wieder eine Art ternäre Funktionskompositionsoperation, die als Ergebnis die Anzahl der Ableitungen am längsten Ableitungspfad zurückgibt.

$F_2 : \Sigma_{BNF} \rightarrow \Sigma_{DCF}$ ist nun wie folgt definiert:

- i. Für alle Elemente aus V_T ist F_2 definiert durch:

$$\forall x \in V_T : x \rightarrow \varphi(x, d - 1)$$

Wir fügen für alle $x \in V_T$ eine Klausel der folgenden Art zur rekursiven Ableitungszählfunktion φ hinzu:

$$\varphi(x, d) = \begin{cases} 0 & \text{wenn } d = 0 \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

- ii. Für alle Elemente aus V_N ist F_2 wie folgt definiert:

$$\forall y_{LHS} \in V_N : y_{LHS} \rightarrow \varphi(y_{LHS}, d)$$

$$\forall y_{RHS} \in V_N : y_{RHS} \rightarrow \varphi(y_{RHS}, r_i)$$

Mit einem Nichtterminalsymbol y und mit $d < 1$ als Argumente ist φ nicht definiert. Für r_i , siehe Ausdruck 6.

- iii.

$$: \rightarrow = s(z)$$

wobei z die gesamte rechte Seite der Produktionsregel ("dem Rest") darstellt. $s(n)$ ist die Nachfolgerfunktion, die durch folgende Regeln definiert wird:

- A. $s(n) = n + 1$,
- B. $s(0) = 1$ und
- C. $s(\text{nicht definiert}) = \text{nicht definiert}$.

$\mapsto \max$

max wird für den Wert *nicht definiert* mit folgenden zusätzlichen Regeln vervollständigt:

- A. $\max(\text{nicht definiert}, \text{nicht definiert}) = \text{nicht definiert}$,
- B. $\max(n_1, \text{nicht definiert}) = n_1$ und
- C. $\max(\text{nicht definiert}, n_2) = n_2$.

$\star \rightarrow \bullet$

Jede k -Symbolkette y^k auf der rechten Seite einer Produktionsregel wird durch einen Funktionsaufruf $\bullet(y^k, d - 1)$ ersetzt. Die $k + 1$ -äre Operation \bullet ist definiert durch:

$$\bullet(y^k, d) = \max_{\sum_{i=1}^k r_i = d, r_i \geq 0, d \geq 0} \sum_{i=1}^k \varphi(y_i, r_i) \tag{6}$$

Die Struktur dieses Ausdrucks entspricht der Struktur von Ausdruck 3.

φ ist eine rekursive Funktion, die das Argument d als Ergebnis zurückgibt, wenn zumindest ein Wort in L_1 in d Ableitungsschritten abgeleitet werden kann. Wenn kein Wort in L_1 in d Ableitungsschritten abgeleitet werden kann, liefert φ den Wert *nicht definiert* als Ergebnis.

(b) Im zweiten Schritt wird φ so geändert, daß die Übersetzung Φ berücksichtigt wird. Wie dies im speziellen Fall durchzuführen ist, hängt von der Beschränkungsrelation Φ ab.

3. Zuletzt wird mittels einer geeigneten Technik versucht zu zeigen, daß $\text{card}(S_{L,d}) < \text{card}(S_{L_2,\Phi(d)})$ fast überall gilt. Für anwendbare Techniken wird der Leser an einen Spezialisten (mit schwarzem Gürtel) für erzeugende Funktionen verwiesen [Wilf, 1990].

V. FRL – Die Regelsprache in der Methode der Boston Consulting Gruppe zur Entwicklung eines strategischen Unternehmensportfolioplans

Die Methode der Boston Consulting Gruppe zur Entwicklung eines strategischen Unternehmensportfolioplans stellt eines der bekanntesten qualitativen Verfahren auf dem Gebiet der strategischen Unternehmensplanung dar [Kotler, 1980]. In diesem Abschnitt wird bewiesen, daß ein unscharfes Classifiersystem mit der unscharfen Regelsprache FRL , die in [Geyer-Schulz, 1993; Bandemer,

1993] präsentiert wird, eine Regelsprache schneller lernt als ein klassisches Classifiersystem mit der Regelsprache RL . Im Gegensatz zur (endlichen) unscharfen Regelsprache FL , die in [Geyer-Schulz, 1992; Lowen, 1992] analysiert wurde, werden die Ausprägungen einer linguistischen Variable in FRL durch eine kontextfreie Grammatik, nicht durch Enumeration, wie in FL , definiert. Die (unendliche) Sprache FRL wird durch eine kontextfreie Grammatik definiert, die die Formulierung ziemlich komplexer Regeln erlaubt. Eine unmittelbare Folge davon ist, daß die Konstruktion der Ableitungsbeschränkungsfunktion ϕ_I aus der Beschränkungsrelation Φ_I nicht mehr so einfach und offensichtlich wie im [Geyer-Schulz, 1992] präsentierten Beispiel ist. Offensichtlich hängt die Komplexität von Lernalgorithmen dennoch hauptsächlich von der Suchraumgröße ab. Alle Theorem beziehen sich deshalb auf den Vergleich von Suchraumgrößen.

Mit Hilfe des Darstellungstheorems von [Negoiita and Ralescu, 1975] kann jeder unscharfen Regelbasis in FRL eine klassische Regelbasis in RL abgeleitet werden, die die unscharfe Regelbasis approximiert. Wenn jedoch keine Annäherung, sondern eine exakte Übersetzung Φ einer unscharfen Regelbasis in FRL in eine klassische Regelbasis RL gefordert wird, so erhält man:

Theorem 2 Die Übersetzung einer Regel in FRL durch Φ erzeugt überabzählbar viele Regeln in R .

Beweis Durch Anwendung des Darstellungstheorems kann eine unscharfe Teilmenge durch eine Familie überabzählbar vieler klassischer Teilmengen dargestellt werden. (Siehe [Negoiita and Ralescu, 1975].) ■

Offensichtlich ist die Übersetzung Φ nicht für Rechnerimplementierungen geeignet, da Φ für jede Regel in FRL eine unendliche Anzahl von Regeln in RL erzeugt. Das bedeutet, daß eine unscharfe Regelsprache einen unendlichen Speed-Up in der Lerngeschwindigkeit im Vergleich zu einer klassischen Regelsprache zumindest theoretisch erreicht. Rechnerimplementierungen von unscharfen Regelsprachen approximieren das reelle Einheitsintervall durch eine endliche Kette I . Trotz dieser Approximation erreichen unscharfe Regelsprachen noch immer einen beträchtlichen Speed-Up im Vergleich zu klassischen Regelsprachen.

Theorem 3 $\text{card}(SFRL,d) < \text{card}(SRL,\Phi_I(d))$, für alle $d \geq k$.

Beweis Auf Grund von Platzbeschränkungen wird der Leser auf [Geyer-Schulz, 1993] für eine vollständigere Version des Beweises verwiesen. Um dennoch zumindest den Ausgangspunkt des Beweises zu zeigen, folgen die im Beweis verwendeten Definitionen von FRL , RL and $\Phi(I)$.

1. Die Grammatik von FRL besteht aus den folgenden Produktionsregeln:
(rule base) ::= (rule) | (rule) (rule base)

$$\forall \alpha \in I : \begin{array}{l} \text{(frame) "USED" "IF" (truth value)} \implies \\ \text{(frame) "USED" "IF" (truth value) "WITH" } (\alpha) \end{array} \quad (7)$$

$$\text{"(noun) "IS" (verbal expression)"} \implies \text{"(noun) "IN" (interval list)"} \quad (8)$$

Bei der Berechnung von Tabelle 1 wurde eine Kette I der Länge 5 verwendet.

Die Größenordnung der Differenz der Suchraumgrößen von FRL und RL ist noch immer unbekannt. Um diese Frage zu beantworten, wurden die ersten 20 Koeffizienten der wie oben beschrieben abgeleiteten Potenzreihen am Rechner ausgewertet. Da es schwierig ist, Differenzgleichungsschemen in eine geschlossene Form zu bringen, wurden die rekursiven Differenzgleichungen direkt ausgewertet. Tabelle 1 enthält die so erhaltenen Resultate. Die Fragezeichen in der letzten Spalte zeigen die Grenzen dieses Ansatzes auf. Trotz einer 12 MB großen virtuellen Maschine mußten die Berechnungen auf Grund von Hauptspeichermangel abgebrochen werden. Die Koeffizienten, die dennoch berechnet werden konnten, zeigen, daß der Suchraum von FRL beträchtlich kleiner als der Suchraum von RL ist.

d	$\phi_I(d)$	$card(S_{FRL,d})$	$card(S_{RL,\phi_I(d)})$
7	50	120	$3.1359 \cdot 10^{444}$
9	50	$1.6800 \cdot 10^3$	$3.1359 \cdot 10^{444}$
11	70	$2.5560 \cdot 10^4$	$1.4307 \cdot 10^{637}$
13	90	$4.4040 \cdot 10^5$	$6.5276 \cdot 10^{829}$
14	100	$4.5480 \cdot 10^5$	$7.5585 \cdot 10^{906}$
15	100	$8.0293 \cdot 10^6$	$7.5585 \cdot 10^{906}$
16	100	$8.4037 \cdot 10^6$	$7.5585 \cdot 10^{906}$
17	110	$1.5221 \cdot 10^8$	$2.5720 \cdot 10^{945}$
18	120	$1.6037 \cdot 10^8$?
19	130	$2.9762 \cdot 10^9$?
20	140	$3.1502 \cdot 10^9$?

Tabelle 1: Die Tiefe der Ableitungsbäume und die Suchraumgrößen in RL und FRL

(rule) := (frame) "USED" "IF" (truth value)
 (truth value) := (truth value) (connective) (truth value) | (" (noun) "IS" (verbal expression) ")
 (verbal expression) := (adjective) | (adverb) (verbal expression) | (" (connective) (verbal expression)
 (adjective) := "HIGH" | "LOW" | "MEDIUM" | "UNDEFINED" | "UNKNOWN"
 (adverb) := "ABOVE" | "BELOW" | "AROUND" | "UPPER" | "LOWER" | "MORE_OR_LESS"
 | "RATHER" | "VERY" | "NOT" | "NEITHER" | "POSSIBLY" | "TRULY" | "FUZZILY"
 (connective) := "AND" | "OR" | "BUT" | "NOR" | "TO" | "EXCEPT"
 (noun) := "MARKETGROWTH" | "RELATIVE_SHARE" | "SALES"
 (frame) := "STAR" | "QUESTION_MARK" | "CASH_COW" | "DOG" | "BUILD" | "HARVEST" | "HOLD" | "DIVEST"

2. Die formale Sprache RL wird durch folgende Produktionsregeln ihrer Grammatik definiert:

(rule base) := (rule) | (rule) (rule base)
 (rule) := (frame) "USED" "IF" (truth value) "WITH" (α)
 (truth value) := (truth value) (connective) (truth value) | (" (noun) "IN" (interval list) ")
 (connective) := "AND" | "OR" | "BUT" | "NOR" | "TO" | "EXCEPT"
 (noun) := "MARKETGROWTH" | "RELATIVE_SHARE" | "SALES"
 (frame) := "STAR" | "QUESTION_MARK" | "CASH_COW" | "DOG" | "BUILD" | "HARVEST" | "HOLD" | "DIVEST"

(interval list) := (interval) | (interval) (interval list)

(interval) := [{" (real) " ; (real) "]"

(real) bezeichnet die m -Bit Darstellung einer Gleitpunktzahl. Im Beweis und für die Berechnung von Tabelle 1 werden Gleitpunktzahlen mit 8 Byte Länge angenommen.

(α) ist eine (real) Zahl in $[0, 1]$.

3. Mit der Hilfe des Darstellungstheorems von [Negoiita and Ralescu, 1975] kann eine Abbildung $\Phi : FRL \mapsto RL$ konstruiert werden, die jede un-scharfe Regel in eine (unendliche) indizierte Familie von Regeln über dem reellen Einheitsintervall $[0, 1]$ übersetzt. Wird jedoch eine endliche Kette I mit Elementen in $[0, 1]$ gewählt, so kann Φ durch $\Phi_I : FRL \mapsto RL$ approximiert werden [Kruise and Meyer, 1987]. Φ_I wird durch die folgende Menge von Metaproduktionsregeln definiert [Geyer-Schulz, 1992]:

Literatur

- [Aho and Ullman, 1972] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling, Volume I: Parsing*, volume 1. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1972.
- [Bandemer, 1993] Hans Bandemer, editor. *Modelling Uncertain Data*, volume 68 of *Mathematical Research*, Berlin, 1993. Akademie Verlag.
- [Blum, 1967] Manuel Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the Association for Computing Machinery*, 14(2):322-336, April 1967.
- [Cutland, 1980] Nigel J. Cutland. *Computability - An Introduction to Recursive Function Theory*. Cambridge University Press, Cambridge, 1980.
- [Feferman et al., 1986] Solomon Feferman, John W. Dawson jr., Stephen C. Kleene, Gregory H. Moore, Robert M. Solovay, and Jean van Heijenoort, editors. *Kurt Gödel - Collected Works, Publications 1929 - 1986*, volume 1, New York, 1986. Oxford University Press.
- [Geyer-Schulz, 1992] Andreas Geyer-Schulz. Fuzzy classifier systems. In Lowen [1992]. To appear.
- [Geyer-Schulz, 1993] Andreas Geyer-Schulz. On the specification of fuzzy data in management. In Bandemer [1993], pages 105-110.
- [Gödel, 1936] Kurt Gödel. Über die Länge von Beweisen. *Ergebnisse eines mathematischen Kolloquiums*, 7:23-24, 1936. Quoted from the reprint in [Gödel, 1986].
- [Gödel, 1986] Kurt Gödel. Über die Länge von Beweisen. In Feferman et al. [1986], pages 396-398.
- [Goulden and Jackson, 1983] Ian P. Goulden and David M. Jackson. *Combinatorial Enumeration*. John Wiley & Sons, New York, 1983.
- [Kotler, 1980] P. Kotler. *Marketing Management - Analysis, Planning and Control*. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.
- [Kruse and Meyer, 1987] Rudolf Kruse and Klaus D. Meyer. *Statistics with Vague Data*. D. Reidel Publishing Company, Dordrecht, 1987.
- [Kuich and Salomaa, 1986] Werner Kuich and Arto Salomaa. *Seminings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin, 1986.
- [Lowen, 1992] Robert Lowen, editor. *Fuzzy Logic: State of the Art*, Dordrecht, 1992. Kluwer Academic Publishers. To appear.
- [Negoiita and Ralescu, 1975] C. V. Negoita and D. A. Ralescu. Representation theorems for fuzzy concepts. *Kybernetes*, 4:169-174, 1975.
- [Parikh, 1973] Rohit J. Parikh. Some results on the length of proofs. *Transactions of the American Mathematical Society*, 177:29-36, March 1973.
- [Salomaa, 1973] Arto Salomaa. *Formal Languages*. ACM Monograph Series. Academic Press, New York, 1973.
- [Salomaa, 1990] Arto Salomaa. Formal languages and power series. In van Leeuwen [1990], pages 103-132.
- [Statman, 1978] Richard Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, 15:225-287, 1978.
- [van Leeuwen, 1990] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science - Formal Models and Semantics*, volume B, Amsterdam, 1990. Elsevier.
- [Wilf, 1990] Herbert S. Wilf. *generatingfunctionology*. Academic Press, San Diego, CA, 1990.
- [Young, 1973] Paul Young. Easy constructions in complexity theory. *Proceedings of the American Mathematical Society*, 37(2):555-563, September 1973.