Über einige lokal verallgemeinernde Speicher und ihre Weiterentwicklung Teil 1: CMAC/AMS und MIAS

One some Locally Generalizing Storage Methods and their further Development Part 1: CMAC/AMS and MIAS

Henning Tolle

Bei der Modellierung, Steuerung und Regelung nichtlinearer Prozesse werden häufig zwei- bzw. mehrdimensionale Kennfelder eingesetzt. Sie können äußerst effektiv in lokal verallgemeinernden Speichern abgelegt werden. Der Beitrag stellt an Hand von drei Beispielen und ihren Weiterentwicklungen dar, dass sehr verschiedenartige Ansätze möglich sind und die Auswahl gemäß ihren inherenten Vor- und Nachteilen von der ins Auge gefassten Anwendung bestimmt werden sollte. Teil I behandelt mit „CMAC/AMS" ein neurobiologisch motiviertes und mit „MIAS" ein aus der grafischen Datenverarbeitung abgeleitetes Verfahren.

For the modelling, feedforward and feedback control of nonlinear processes one uses frequently two- and/or multidimensional characteristics. Such characteristics can be represented effectively by locally generalizing storage methods. By three different examples and their further developments it is shown in this paper, that very different approaches are possible and that taking into account the inherent advantages and disadvantages one should choose the most appropriate method according to the envisaged application. Part I deals through "CMAC/AMS" with a neurobiologically motivated technique and through "MIAS" with one derived from the area of computer graphics.

Schlagwörter: Kernfeldspeicherung, CMAC, lokal verallgemeinernde Speicherung, nichtlineare Regelung, lernende Regelung

Keywords: Storage of characteristics, CMAC, locally generalizing storage, nonlinear control, learning control

1 Einleitung

Die hohe Leistungsfähigkeit preiswerter Rechentechnik erlaubt es heute, bei der Steuerung und Regelung technischer Prozesse nichtlineare Effekte im Prozess zu berücksichtigen und über nichtlineare Ansätze Verbesserungen gegenüber einer rein linearen Synthese zu erzielen. Allerdings sind die dafür notwendige detaillierte Prozessmodellierung und die Erarbeitung geeigneter Korrekturstrategien im Allgemeinen schwierig oder sehr aufwändig. Bei nicht nur projektierten, sondern bereits vorhandenen Anlagen kann man zur Lösung dieses Problems von Messdaten ausgehen und die Steuerung bzw. Regelung mit Hilfe von aus den Messdaten abgeleiteten Kennfeldern gestalten. Diese können über abgespeicherte Daten – off-line – oder während des Betriebs – on-line – generiert werden, wobei der Off-line-Ansatz eine ausreichende Vorab-Datenaufnahme voraussetzt, während der On-line-Ansatz lernend die Regelung/Steuerung aufbaut bzw. eine vorhandene Regelung/Steuerung zu verbessern erlaubt. In beiden Fällen muss bei der praktischen Realisierung durch eine Überwachungsebene oder andere Maßnahmen sichergestellt werden, dass keine Anlagengefährdung durch eine noch nicht erprobte Stellaktion auftritt.

Für die Verknüpfung von Situationsdaten und zugehörigen Reaktionen wird vielfach von ,,künstlichen neuronalen Netzen – KNN" ausgegangen, die eine sehr vereinfachte Imitation der Vorgänge in Nervennetzen darstellen. Besonders beliebt ist das in Bild 1 skizzierte ,,Multilayer Perceptron", bei dem eine Überlagerung von Sigmoid-Funktionen auf die Eingangs-Ausgangszusammenhänge trainiert und im Rechner für einen späteren Abruf gespeichert wird.

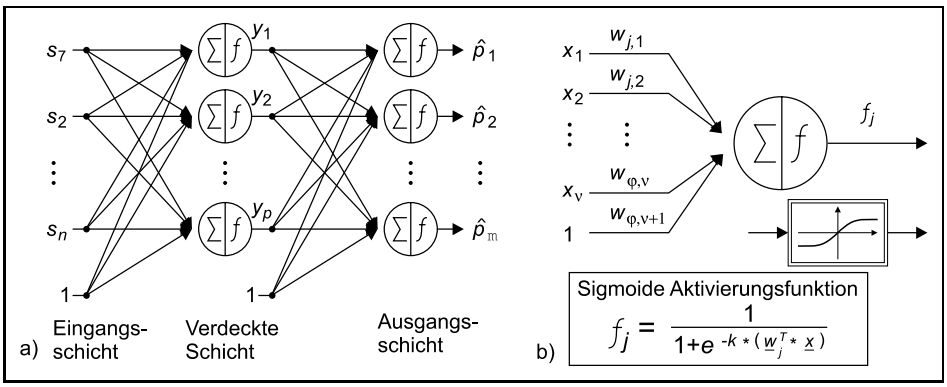


Bild 1: a) Multilayer Perceptron mit einer verdeckten Schicht; b) Modellneuron, global interpolierend – $w_{j,i}$ einstellbare Gewichte in der zugehörigen sigmoiden Aktivierungsfunktion; der Wert 1 in a) ergibt einen Gleichanteil – aus [8].

Ich möchte im Folgenden darauf aufmerksam machen, dass es vielfältige Werkzeuge gibt, um solche Zusammenhänge darzustellen, die allgemein als mehrdimensionale Kennfelder anzusehen sind, und dass man die Werkzeuge auf die Anwendung zuschneiden sollte.

Dazu will ich einige Entwicklungen meiner Arbeitsgruppe benutzen, die im Wesentlichen nur in den im Allgemeinen selten im Detail gelesenen Dissertationen zu finden sind.

Vorab sei aber darauf hingewiesen, dass man meines Erachtens zwei Aufgabenstellungen unterscheiden sollte, für die „neuronale Netze“ oder allgemeiner Kennfelder herangezogen werden können: Die Klassifikation von Daten und die Nachbildung von Funktionen.

Im ersten Fall bildet das Kennfeld eine Funktion, die Daten trennt, die zu unterschiedlichen Klassen gehören und bei der es nur in den Gebieten, in denen Daten verschiedener Klassen sehr eng beieinander liegen, darauf ankommt, wie das Kennfeld im Detail ausgeformt ist. Insbesondere bei hohen Eingangsdimensionen ist damit die genaue Form des Kennfeldes wenig kritisch. Bei solchen Problemstellungen kann man mit global verallgemeinernden Speichern arbeiten, das sind Speicher, bei denen die Stützfunktionen, aus deren Überlagerung das Kennfeld entsteht, im ganzen Arbeitsraum definiert sind, wie z. B. die Sigmoidfunktionen des „Multilayer Perceptrons“.

Will man hingegen eine Funktion möglichst genau nachbilden, z. B. ein nichtlineares Kennfeld eines Motors, so haben lokal definierte Stützfunktionen einen Vorteil: Bei global definierten Stützfunktionen beeinflusst jeder neue Datensatz nicht nur das Verhalten der Stützfunktionen am Ort des Datensatzes, sondern auch in beliebiger Entfernung davon, d. h. auch jeder vorher an anderer Stelle gut trainierte Zusammenhang wird damit verändert. Als Folge entsteht ein hoher Trainingsaufwand, bis alle Stützfunktionen so austariert sind, dass der nichtlineare Zusammenhang im Rahmen der damit erzielbaren Approximationsgüte ausreichend genau dargestellt ist. Zudem geben globale Stützfunktionen auch an den Stellen eine Antwort auf Eingangsdaten, an denen überhaupt noch keine Zusammenhänge in den Speicher eingegeben worden sind, sie extrapolieren in beliebiger Weise. Dies kann mit nur lokal, d. h. über einem kleinen Gebiet definierten Stützfunktionen, vermieden werden: Sie verändern für jeden neuen Datensatz die Form des Kennfeldes nur in ihrem lokalen Einzugsbereich und geben in untrainiertem Gebiet gar keine Antwort, sodass über einen „Trainingsindikator“ angezeigt werden kann, wo über ausreichend viele eingegebene Datensätze eine vernünftige Antwort des Speichers zu erwarten ist und wo nicht, was z. B. für eine Prozessführung große Bedeutung hat.

Eine Zwischenlösung zwischen beiden Ansätzen stellen „KNN – RBF“, KNN mit „Radial Basis Functions“ dar, bei denen die Sigmoidfunktionen durch Gaußglocken ersetzt werden, die einen fast lokalen Charakter haben. Auf „Trainingsindikatoren“ wird dabei aber im Allgemeinen leider verzichtet.

Bei der Konzentration auf „lokal verallgemeinernde Speicher“ in diesem Aufsatz sollen drei Ansätze mit jüngeren Verbesserungen und ihre Anwendungsbereiche erörtert werden.

Teil I behandelt zwei Verfahren:

Ein neuronal inspirierter Ansatz – CMAC/AMS –, der für die On-line-Anwendung in schnellen Regelkreisen günstig ist, wird in Abschnitt 2 skizziert und mit einer Modifikation ergänzt, die den Zielkonflikt zwischen schnellem Informationsaufbau über dem Eingangsbereich und guter lokaler Konvergenz entspannt. Dies kann als ein einfacher Ansatz zum Umgang mit stark wechselnden Datendichten im Eingangsraum angesehen werden.

Ein aus der grafischen Datenverarbeitung abgeleiteter Ansatz – MIAS –, der für langsamere Regelkreise, wie sie z. B. in chemischen Anlagen vorkommen, geeignet ist und eine weitere Verbesserung der Anpassung an die Datendichte erlaubt und bei dem Probleme der Extrapolation erörtert werden, diskutiert Abschnitt 3.

Ein wertender Vergleich, der auch das Multilayer Perceptron beinhaltet, schließt diesen Teil I ab. In Teil II werden Ideen der Freiflächenapproximation aus der Mathematik – Delaunay-Triangulierungen – benutzt, um niedrigdimensionale Kennfelder mit möglichst wenig Speicherplatz und Rechenaufwandsanforderungen – z. B. für das Motormanagement im Kraftfahrzeug – bereitzustellen.

In der Arbeit, die am Ende von Teil II systematisch zusammengefasst ist, wird versucht, die wesentlichen Gedankengänge herauszuarbeiten. Auf Einzelheiten der Implementierungen wird ebensowenig eingegangen, wie auf die Vielfältigkeit der Ansätze für KNN und ihre Verflechtung mit

anderen Methoden. Es sei nur darauf hingewiesen, dass es z. B. eine enge Verwandtschaft zwischen RBF, CMAC und der Fuzzy-Methodik gibt, wie Brown und Harris in [4] zeigen. Übersichten über KNN findet man u. a. in [9; 10; 19].

2 Weiterentwicklung des CMAC/AMS

CMAC ist eine Abkürzung für „Cerebellar Model Articulation Controller“ und wurde von J.S. Albus in [2] als lernendes Kennfeld in Anlehnung an die in seiner Dissertation [3] entwickelten Vorstellungen über Aufgabe und Funktionsweise des menschlichen Kleinhirns vorgestellt. Die Grundidee ist eine Dateninterpolation durch verteilte Speicherung eines Ausgangsmesswertes p über den Eingangswerten s_i $(i = 1, 2, \ldots n)$. Es wird von der realistischen Annahme ausgegangen, dass die n Eingänge jeweils nur bis auf ein ε_i genau, also diskretisiert sind. Dann wird statt einer Rasterung des Eingangsraums mit den Werten ε_i und Ablage des Wertes p an der durch die s_i markierten Stelle, eine Vervielfachung des Eingangsraums um einen Faktor ρ und eine Ablage von $1/\rho \cdot p$ in den jeweils um ε_i gegeneinander verschobenen Grobrasterungen $\rho \cdot \varepsilon_i$ der ρ Eingangsräume vorgenommen. Dies ergibt statt einer Säule der Höhe p für den Eingangsvektor $\underline{s}$ bei Aufaddition der Werte aus den Grobrasterungen eine Stufenpyramide um $\underline{s}$ mit dem richtigen Wert von p im Zentrum. Bild 2 zeigt für das Beispiel eines dynamischen Systems $y(k+1) = f(y(k), u(k))$

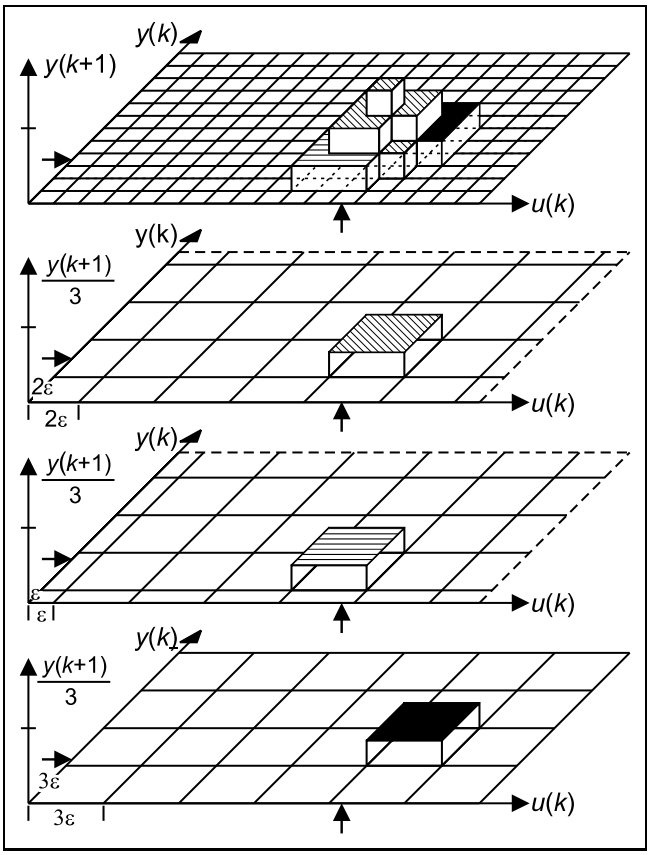


Bild 2: CMAC-Prinzip für 2 Eingänge, einen Ausgang und $\rho = 3$: $y\,(k+1) = f\,(y(k), u(k))$.

mit $p = y(k+1)$, $\underline{s} = (y(k), u(k))$ und $\rho = 3$ die gegeneinander verschobenen 3 Grobrasterungen mit dem Wert $p/3$ und die durch Aufaddition im Ein/Ausgangsraum entstehende Stufenpyramide.

Neben dem Interpolationseffekt durch die lokale Verallgemeinerung des Eingangsbereichs ergibt sich bei diesem Vorgehen auch eine Reduktion des notwendigen Speicherplatzes gegenüber einer reinen Tabellierung. Bei CMAC hat man nach [16] einen maximalen Speicherbedarf von $\rho \cdot \left(\frac{2^b-1}{\rho} + 2\right)^n$ mit b als Auflösung in Bit. Daraus folgt z. B. für $n = 4, b = 8$ und einem Eingangsraum mit ca. $4{,}3 \cdot 10^9$ Punkten, eine Speicherreduktion auf $1{,}34 \cdot 10^6$ Speicherzellen für $\rho = 16$ und auf nur 210 000 Speicherzellen für $\rho = 32$. Da im Allgemeinen nur ein Teil des möglichen Eingangsraums für eine konkrete Anwendung gebraucht wird, ist eine weitere Reduktion durch eine inhaltsadressierte Speicherung (Hashcoding, für eine nähere Beschreibung siehe z. B. [11]) möglich. Dabei ist wegen der verteilten Speicherung eine gewisse Resistenz der Ausgangswerte gegen eine teilweise Zerstörung des Speichers als zusätzlicher Gewinn zu verbuchen. Zudem ist durch die inhaltsadressierte Speicherung die Zugriffszeit fast unabhängig von der Menge der gespeicherten Daten.

Für den in [5] von E. Ersü präsentierten Regelkreis zum On-line-Erlernen einer günstigen nichtlinearen Regelung für existierende, aber schlecht modellierbare Strecken – siehe später Bild 6 – wurde unter seiner Betreuung von J. Militzer 1981 eine effektive Implementierung des Albus'-schen Ansatzes vorgenommen, die als zusätzliches Element einen „Trainingsindikator“ enthält und unter der Bezeichung AMS (Associative Memory System) in [6] erstmalig beschrieben wurde. In [23] findet sich eine ausführliche Dokumentation, eine Methode zur Unterdrückung weißen Rauschens, eine kurze Skizzierung der Konvergenzbeweise für CMAC/AMS und die Ergebnisdarstellung vielfältiger Untersuchungen der Eigenschaften des AMS und seines Einsatzes in lernenden Regelkreisen.

Hier soll nur auf die Grundvorgehensweise kurz eingegangen werden, um die zu besprechende Weiterentwicklung verständlich zu machen. Der durch Aufaddition der Werte in den ρ Grobrasterungen sich für einen Eingangswert ergebende Ausgangswert – bei mehreren Ausgängen erfolgen Speicherungen pro Ausgang – wird durch eine Wertanwesenheitskontrolle in einem zusätzlichen Speicher ergänzt, die bei bereits abgelegtem Wert an der abgefragten Stelle $\alpha = 1$, sonst $\alpha = 0$ ist. Damit ergibt sich für den ermittelten Ausgangswert $\hat{p}$:

$$\hat{p} = \frac{\sum_{j=0}^{\rho-1} \alpha_j \cdot w_j}{\sum_{j=0}^{\rho-1} \alpha_j} \tag{1}$$

mit dem Trainingsindikator

$$\tau = \frac{1}{\rho} \sum_{j=0}^{\rho-1} \alpha_j \cdot 100[\%]\,, \tag{2}$$

der anzeigt, in wie vielen der zugehörigen Rasterelemente Werte für die zur Diskussion stehende Stelle abgelegt sind. Im Allgemeinen wird bei einem $\tau > 70\%–80\%$ davon ausgegangen, dass die zugehörige Region ausreichend trainiert, der abgefragte Ausgangswert also verwendbar ist. Dies ist eine rein geometrische Aussage. Vertiefend kann man, statt nur für das Training an sich $\alpha = 1$ zu setzen, die Häufigkeit des Trainings in einem Rasterelement mit hochzählen und z. B. zusätzlich zu $\tau > 70\%–80\%$ eine minimal durchschnittliche Trainingshäufigkeit bei den trainierten Rasterelementen verlangen.

Diese Trainingsindikation ist sehr einfach und damit für das On-line-Lernen am Prozess besonders geeignet. Kompliziertere Ansätze sind möglich. So findet man z. B. in [21] eine umfangreiche Gültigkeitsanalyse, um den Zuständigkeitsbereich eines off-line erlernten Prozessmodells zu bestimmen.

Im Training werden die Gewichte im Speicher so eingestellt, dass der Wert im neuen Zeitpunkt $k+1$ aus dem Wert im Zeitpunkt k gewonnen wird über

$$w_j^{(k+1)} = w_j^{(k)} + \Delta w_j^{(k)} = w_j^{(k)} + p^{(k)} - \hat{p}^{(k)}$$
$$\text{mit } w_j^{(k)} = 0 \quad \text{für } \alpha_j^{(k)} = 0\,, \tag{3}$$

wobei $\hat{p}^{(k)}$ der Ausgangswert vor dem Trainingsschritt ist und $p^{(k)}$ der in dem Trainingsschritt angefallene Sollwert.

Ein gewisses Problem stellt die Größe der Verallgemeinerung dar, da sie über die mögliche Steilheit der Flanken des zu speichernden Kennfeldes entscheidet. Ist ρ zu groß, können steile Flanken nicht realisiert werden, wird ρ vorsichtshalber sehr klein gewählt, wird der Trainingsaufwand unnötig groß. Bild 3 zeigt dies für die Funktion:

$$p(s_1, s_2) = \sin^2(2\pi s_1) \cdot \sin^2(2\pi s_2) \tag{4}$$

bei einem Diskretisierungsnetz von 65 500 Punkten. Mit $\rho = 8$ sind 10 000 Punkte für eine gute Kennfeldnäherung notwendig, mit $\rho = 16$ nur 1000 Punkte, das sind

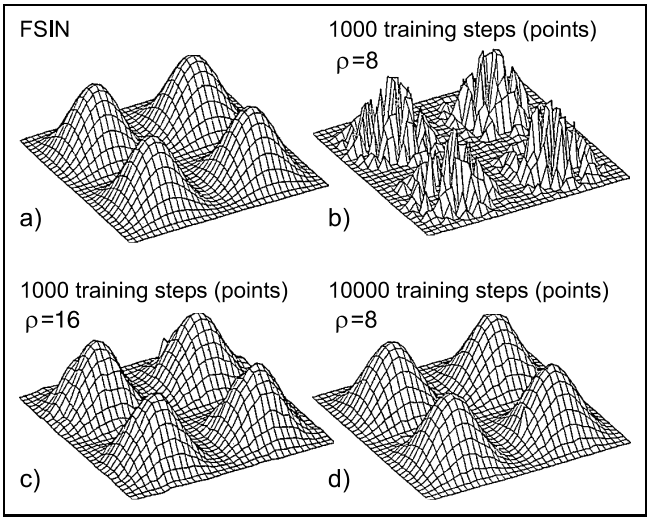


Bild 3: Auswirkung der Größe der Verallgemeinerung ρ für die Funktion (4) – nach [23].

1,5% des bei einer punktweisen Darstellung notwendigen Aufwands. Der mittlere Fehler, der für $\rho = 16$ mit 2500 Trainingspunkten ungefähr bei 5% liegt, beträgt im Übrigen für $\rho = 128$ und 2500 Punkte ca. 25% und ist auch für 20 000 Punkte nicht unter 15% absenkbar, d. h. $\rho = 128$ ist für die vorliegende Funktion zu groß, die auftretenden Flanken können damit nicht dargestellt werden.

Setzt man nicht auf vorab aufgenommenen Messwerten auf, aus denen man experimentell das günstigste ρ ermitteln kann, sondern erlernt man das Kennfeld parallel zur Regelung der Anlage, steckt man somit in einem Dilemma: ein zu großes ρ ist ungeeignet, ein zu vorsichtiges kleines ρ erfordert einen sehr großen Trainingsaufwand, was insbesondere bei hohen Eingangsdimensionen des Kennfeldes von Bedeutung ist. Diese treten bei dynamischen Systemen häufig auf, da dort neben den Zuständen $\underline{x}(t)$ auch $\underline{x}(t-T)$, $\underline{x}(t-2T)$ – T ist die Abtastzeit – im Allgemeinen von Bedeutung sind. Eine mögliche Lösung wurde schon in [23] angegeben, die variable Verallgemeinerung, in der die Messdaten parallel in mehrere Speicher mit unterschiedlich großem ρ eintrainiert und die Ausgangswerte aus demjenigen Speicher entnommen werden, der das kleinste ρ hat, für den der Trainingsindikator noch eine ausreichend sichere Antwort aufzeigt.

W.S. Mischo hat 1991 eine alternative, weniger Speicherplatz verbrauchende Strategie vorgeschlagen, die in [15] dokumentiert ist und auf die ich hier als Weiterentwicklung hinweisen möchte. Er nutzt dazu zwei Gedankengänge: Die Modifikation des Assoziationsalgorithmus für den CMAC/AMS, die von Parks und Militzer in [20] vorgeschlagen wurde und konzeptbedingte Asymmetrien beseitigt – vgl. Bild 4 – und die Idee, die rein diskrete Intervallbetrachtung von Albus mit der Idee der kontinuierlichen rezeptiven Felder zu überlagern, ein Gedanke, der schon in [1; 17] angeführt, aber dort im Wesentlichen nur theoretisch diskutiert worden ist.

Sieht man nämlich den diskreten Wirkungsbereich eines Eingangswertes, das zugehörige Raster oder allgemeiner Hyperwürfel-Gebiet als rezeptives Feld, so ist zu erwarten, dass der Ausgangswert nicht unabhängig von der Lage des Eingangspunktes darin, sondern abhängig von dieser Lage ist.

Um diese Grundidee in eine algorithmische Vorschrift umzusetzen ohne zusätzliche aufwändige Berechnungen, wird von W.S. Mischo der Abstand des Eingangspunktes $\underline{s}$ vom Zentrum jeder Assoziationszelle in der L_1–Norm genutzt:

$$d_j = \sum_{i=1}^{n} d_{ij} \quad n = \text{Eingangsvektordimension} \tag{5}$$
$$(j = 0, 1 \ldots \rho - 1)$$

mit (ρ gerade):

$$d_{ij} = \begin{cases} \left| (s_i)_\rho - (j-1) - \dfrac{\rho}{2} \right| & \text{falls } (s_i)_\rho > (j-1) \\ \left| (s_i)_\rho - (j-1) + \dfrac{\rho}{2} \right| & \text{sonst} \end{cases}$$

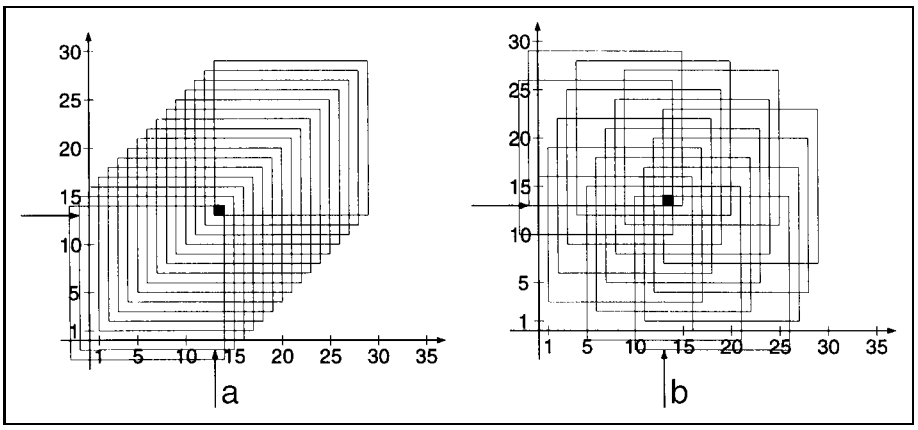


Bild 4: Modifikation der konzeptbedingten Asymmetrien der CMAC/AMS-Assoziation durch [20] für das Beispiel $\rho = 16$, 2 Eingänge: a) Extremfall der Grobrasterung nach dem ursprünglichen Algorithmus, b) Grobrasterung nach dem Vorschlag von Parks/ Militzer – nach [16].

und

$$(s_i)_\rho = \text{ganzzahliger Rest von } \frac{s_i}{\rho} \text{ (Modulo-Operation)}$$

Damit kann dann die Aktivierung der Assoziationszelle angegeben werden, die im Prinzip durch jede achsensymmetrische Funktion mit Maximum bei Null darstellbar ist, z. B. eine auf die Assoziationszelle lokal begrenzte (mehrdimensionale) Gaußglocke – vgl. auch Bild 5 –

$$\varphi_j\left(d_j\right) = \mathrm{e}^{-K\left(d_j/d_{\max}\right)^2} \tag{6}$$

mit $d_{\max} = \frac{n\cdot\rho}{2}$ für ρ gerade als Normierung und einer freien Konstante K zur Anpassung an numerische Erfordernisse.

Die Regel (1) für die Ausgangswertbildung modifiziert sich zu:

$$\hat{p} = \frac{\sum\limits_{j=0}^{\rho-1} \alpha_j \varphi_j w_j}{\sum\limits_{j=0}^{\rho-1} \alpha_j \varphi_j} \tag{7}$$

und die Regel (3) für das Training zu:

$$w_j^{(k+1)} = w_j^{(k)} + \frac{\varphi_j^{(k)} \sum\limits_{j=0}^{\rho-1} \varphi_j^{(k)}}{\sum\limits_{j=0}^{\rho-1} \left(\varphi_j^{(k)}\right)^2} \cdot \left(p^{(k)} - \tilde{p}^{(k)}\right) \tag{8}$$

mit

$$w_j^{(k)} = p^{(k)} \quad \text{für} \quad \alpha_j^{(k)} = 0$$

und

$$\tilde{p}(k) = \frac{\sum\limits_{j=0}^{\rho-1} \left[\alpha_j^{(k)} \varphi_j^{(k)} w_j^{(k)} + \left(1 - \alpha_j^{(k)}\right) \varphi_j^{(k)} p^{(k)}\right]}{\sum\limits_{j=0}^{\rho-1} \varphi_j^{(k)}}$$

Für weitere Einzelheiten vergleiche man [16].

Das Verfahren führt zu der gewünschten besseren Konvergenz für höhere Verallgemeinerungen (typisch: $\rho = 100$). Damit ergibt sich die Möglichkeit neuer Strategien der Parametrisierung für CMAC/AMS. Gilt sonst, dass nur kleine Verallgemeinerungen bei Funktionen mit starken Krümmungen eine gute Approximation ergeben, wird diese Tendenz nun erheblich abgeschwächt. Praktische Versuche zeigten laut [14], dass z. B. mit $\rho = 128$ eine ähnlich feine

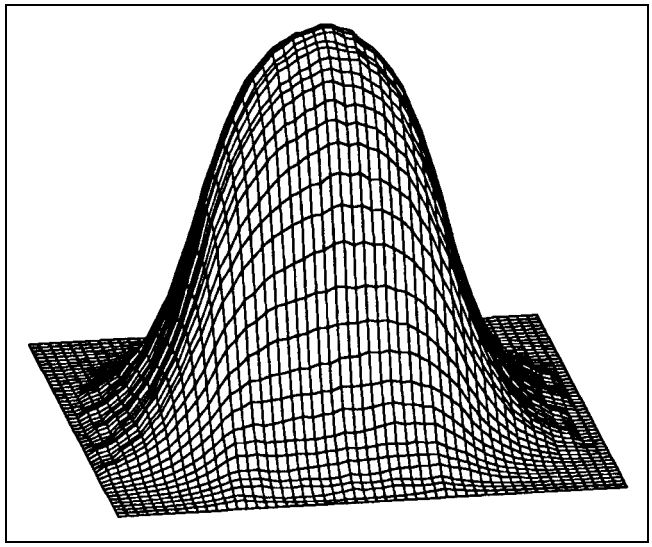


Bild 5: Gaußglocke als rezeptives Feld für das Beispiel von 2 Eingängen – aus [15].

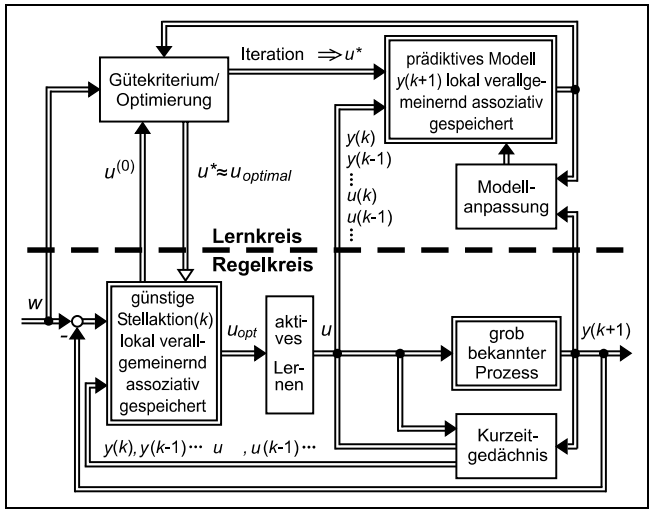


Bild 6: Lernender Regelkreis nach E. Ersü aus [22].

Auflösung erreicht wird wie bei CMAC/AMS sonst bei $\rho = 16$. Hohe Verallgemeinerungen helfen aber im Allgemeinen im Anfangsstadium des On-line-Lernens.

Der Nutzen des neuen Ansatzes sei an einem einfachen Simulationsbeispiel für das On-line-Lernen in einem lernenden Regelkreis nach [14] gezeigt:

Bild 6 zeigt dazu die Struktur des lernenden Regelkreises nach E. Ersü, deren nähere Erläuterung man z. B. in [23] findet. Grob bekannter Prozess heißt, dass man eine vernünftige Abtastzeit T abschätzen kann und dass man weiß, wie viel Vergangenheitsschritte außer $y(k), u(k)$, also zum Zeitpunkt $k \cdot T$, man für eine ausreichende Erfassung der Dynamik des Prozesses benötigt, d. h. wie weit man bei $y(k-1), y(k-2)$ etc. zu gehen hat.

Der im vorliegenden Fall zu regelnde Prozess war aus Testzwecken ein einfacher diskreter linearer Regelkreis mit der Abtastzeit $T = 1$:

$$G(z) = \frac{0{,}01867z^{-1} + 0{,}01746z^{-2}}{1 - 1{,}7826z^{-1} + 0{,}8187z^{-2}} \qquad (9)$$

und der Sprungantwort aus Bild 7. Die Aufgabe war das Verfolgen einer Rampe. Da hier nicht der lernende Regelkreis, sondern nur der neue AMS-Ansatz von Bedeutung ist, wird auf die Parametrisierung des lernenden Regelkreises (Optimierungskriterium, benutzte Vergangenheit) nicht näher eingegangen, sondern es wird nur in Bild 8 das Folgeverhalten nach 10 und 30 Trainingsläufen bei der Verwendung von AMS-Speichern mit dem alten Algorithmus und Verallgemeinerungsansatz für $\rho = 16$ und $\rho = 50$ mit dem neuen Verallgemeinerungsansatz für $\rho = 50$ verglichen. Da die Strecke linear ist, ist eine relativ große Verallgemeinerung unproblematisch und nur von Vorteil, wie man am Vergleich der Fälle $\rho = 16$, $\rho = 50$ – bisheriger Algorithmus – sieht. Dennoch weist, wie der Vergleich $\rho = 50$, bisheriger und neuer Algorithmus zeigt, der Übergang zu dem flexibleren Ansatz von W.S. Mischo, der die Diskretisierung mit einem rezeptiven Feld und einer entsprechenden Abstandsgewichtung im betrachteten Einflussgebiet überlagert, Lernvorteile auf. Dies würde sich bei nichtlinearen Prozessen noch deutlicher zeigen, was aber wegen der Schwerpunktsetzung der Dissertation von W.S. Mischo auf eine Chip-Implementierung des AMS nicht weiter verfolgt wurde: Die Zugriffszeit für

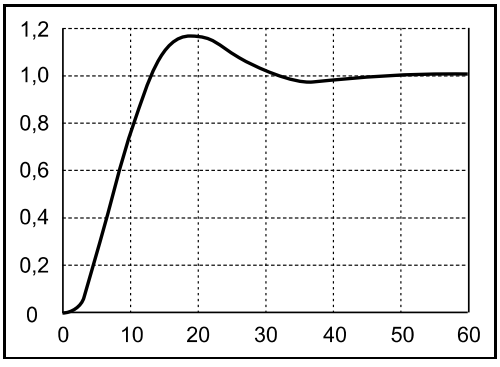


Bild 7: Sprungantwort von (9) – aus [14].

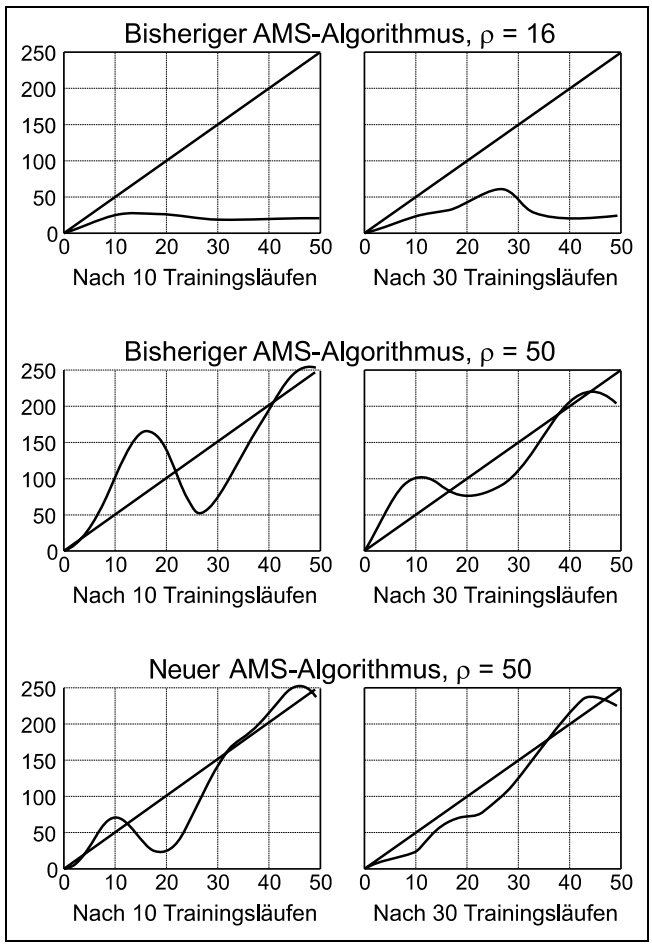


Bild 8: Folgeverhaltensvergleich für (9) bei Einsatz des lernenden Regelkreises – aus [15].

eine AMS-Assoziation kann mit einer solchen Hardware-Implementierung von ca. 1 msec um etwa den Faktor 100 verringert werden.

3 Weiterentwicklung von MIAS

Mit ρ wird beim CMAC/AMS vorab ein lokaler Verallgemeinerungsbereich festgelegt. Kann man stattdessen diesen Bereich selbsttätig an die Dichte der Daten anpassen?

J. Militzer hat dazu 1986 ein Konzept entwickelt, das nicht an Vorstellungen vom menschlichen Gehirn orientiert ist, sondern sich auf mathematische Ansätze zur Konstruktion glatter Flächen in der Computergrafik abstützt. Er kombinierte Ideen von D.H. McLain [13] und R.J. Renka/A.K. Cline [18], nannte seine Methodik MIAS = McLain-type Interpolating Associative Memory und schlug folgendes Vorgehen vor ([12], vgl. auch 23]):

Man legt alle bekannten Ein-Ausgangswertpaare $(\tilde{\underline{s}}_i, \tilde{p}_i)$ in einer Datei ab. Für einen Eingangswert $\underline{s}$, zu dem der Ausgangswert mit Hilfe der bekannten Daten geschätzt werden soll, bestimmt man die Abstände von $\underline{s}$ zu allen $\tilde{\underline{s}}_i$, und numeriert diese so um, dass die Abstände der Größe nach geordnet sind:

$$\tilde{d}_1(\underline{s}) \equiv |\tilde{\underline{s}} - \tilde{\underline{s}}_1| \leq \tilde{d}_2(\underline{s}) \leq \tilde{d}_3(\underline{s}) \ldots \qquad (10)$$

Als Einflussbereich, aus dem Daten benutzt werden (Verallgemeinerungsbereich), wird eine Zahl $q+1$ festgelegt, womit sich eine Hyperkugel im n-dimensionalen Eingangsraum mit dem Radius $\tilde{d}_{q+1}(\underline{s})$ ergibt. Bild 9 zeigt dies für das Beispiel $n=2$ und $q=6$.

Die Schätzung $\hat{p}$ für den Ausgangswert p zu $\underline{s}$ erhält man über eine wie folgt optimierte Hyperebene:

$$g(\underline{x}) = a_0 + \sum_{k=1}^{n} a_k x_k \quad \text{mit } \underline{a} \text{ aus} \tag{11}$$

$$J = \sum_{r=1}^{q} \tilde{\omega}_r \left[g(\underline{\tilde{s}}_r - \underline{s}) - \tilde{p}_r \right]^2 \Rightarrow \underset{\underline{a}}{\text{Min}}$$

und den Gewichtungsfaktoren:

$$\tilde{\omega}_r = \left[\frac{1}{\tilde{d}_r(\underline{s})} - \frac{1}{\tilde{d}_{q+1}(\underline{s})} \right]^2 .$$

Die Gewichtungsfaktoren $\tilde{\omega}_r$ sorgen dabei dafür, dass die Werte nahe bei $(\underline{s})$ stärker berücksichtigt werden als die entfernteren Werte. Der gesuchte Wert $\hat{p}$ ist dann gegeben durch $g(\underline{o}) = a_o = \hat{p}$.

Der hier beschriebene Algorithmus führt zu einer eindeutigen Lösung, wenn die q Stützpunkte den Eingangsraum voll aufspannen. Überlegungen, wie der Algorithmus in entarteten Fällen erweitert werden kann, finden sich in [12].

Man muss im Übrigen natürlich beachten, dass bei allgemeinen nichtlinearen Hyperflächen, wie sie z. B. für $n=2$ in Bild 3 gezeigt sind, vom interessierenden Punkt weit entfernt liegende Punkte ungeeignet sind, die Schätzung zu stützen. Hier ist also ein maximaler Radius $\tilde{d}_{q+1}(\underline{s})$ festzulegen, für dessen Festlegung es bei unbekannten Hyperflächen allerdings keinen systematischen Ansatz gibt. Um einen mit dem AMS-Ansatz kompatiblen Trainingsindikator zu erhalten, benutzt J. Militzer allerdings nicht direkt den maximalen Radius, sondern einen Kehrwert, womit als Indikator folgt:

$$\tau' = \frac{1}{1+\chi \cdot \tilde{d}_{q+1}(\underline{s})} \cdot 100[\%] , \tag{12}$$

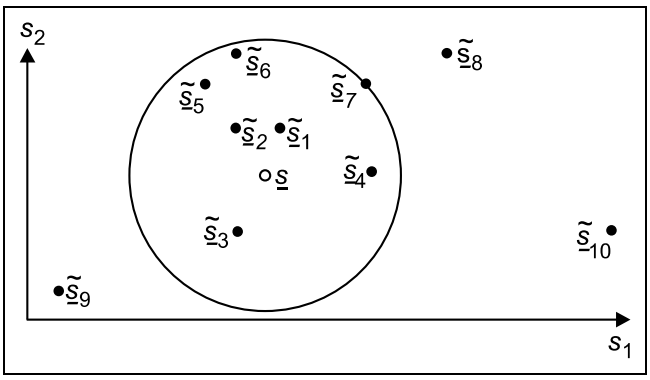


Bild 9: Beispiel für den genutzten Datenbereich zur Schätzung des Ausgangswerts bei bekannten, zufällig gestreuten Datenpaaren $(\tilde{\underline{s}}_i\,,\,\tilde{p})$ beim Ansatz MIAS ($n=2$, $q=6$).

wobei von einer geeigneten Normierung der Abstände auszugehen ist, um sie dimensionslos zu machen oder χ mit der Dimension 1/Länge versehen werden muss; bewährt hat sich $\chi = 1/200$.

Die von S. Gehlen in [8] beschriebenen Modifikationen von MIAS gehen in zwei Richtungen:

Eine stark unsymmetrische Konzentration der Stützstellen am Rand des Einflussgebietes führt zur Extrapolation statt zur Interpolation, was durch einen weiteren Trainingsindikator $\bar{\tau}'$ erfasst werden kann (vgl. Bild 10).

S. Gehlen berechnet dazu den Schwerpunkt $\underline{s}^{sp}$ der betrachteten q Stützstellen:

$$s_k^{sp} = \frac{1}{q} \cdot \sum_{i=1}^{q} s_{i,k} \quad k = 1, 2-n$$

$$\underline{s}^{sp} = \left(s_1^{sp}, s_2^{sp}, \ldots, s_n^{sp}\right)^T ; \tag{13}$$

und den Abstand des Schwerpunktes $\underline{s}^{sp}$ von $\underline{s}$:

$$d^{sp}(\underline{s}) = |\underline{s} - \underline{s}^{sp}| = \sqrt{\left(s_1 - s_1^{sp}\right)^2 + \ldots \left(s_n - s_n^{sp}\right)^2} , \tag{14}$$

Als den die Unsymmetrie beschreibenden ergänzenden Trainingsindikator definiert er dann:

$$\bar{\tau}' = 1 - \frac{d^{sp}(\underline{s})}{\tilde{d}_{q+1}(\underline{s})} , \tag{15}$$

d. h. er nutzt dazu die durch die Größe des Einflussradius normierte Schwerpunktslage.

Die zweite Modifikation von S. Gehlen beschäftigt sich mit einem inherenten Unterschied von CMAC/AMS und MIAS. Während beim CMAC/AMS durch die inhaltskodierte Speicherplatzfestlegung die Zugriffszeit im Prinzip unabhängig von der gespeicherten Zahl von Trainingswerten ist, wächst bei MIAS durch die notwendige Durchsuchung aller Daten für die Abstandsberechnung der Rechenaufwand mit der Zahl der Trainingswerte, was durch geeignete Suchverfahren gemildert – vgl. z. B. [7] – aber nicht prinzipiell beseitigt werden und damit insbesondere beim On-line-Lernen Probleme aufwerfen kann. Diese Schwierigkeit kann verringert werden, wenn man nahe beieinander liegende Werte nicht als verschiedene Punkte, sondern zufällige Messpunktsstreuungen auffasst und sie zusammenfasst. Dies geschieht nach S. Gehlen, indem für neue Trainingswerte der Abstand zu den alten Eingangswerten berechnet wird und nur solche Trainingspaare $(\tilde{\underline{s}}_i, \tilde{p}_i)$ neue Trainingspunkte ergeben, die von allen alten Trainingspaaren einen Abstand größer einem $R_{\min}$ im Eingangsraum haben. Sonst wird nur die bereits gespeicherte Stützstelle i im Ausgangswert modifiziert zu:

$$\tilde{p}_i^{\,\text{neu}} = \frac{\tilde{p}_j + \kappa \cdot \tilde{p}_i^{\,\text{alt}}}{\kappa + 1} , \tag{16}$$

wobei κ einen Zähler kennzeichnet, der in jedem Trainingsschritt der Zelle inkrementiert wird. Bei zeitvarianten Prozessen sollten neue Informationen stärker als die alten

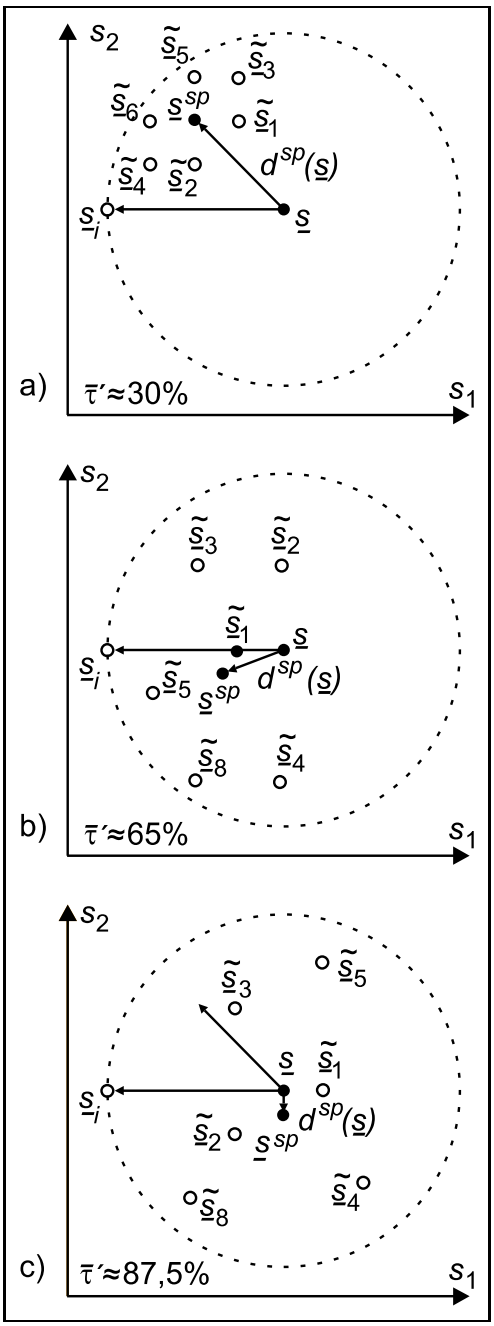


Bild 10: Beurteilung der Stützstellenkonfiguration durch den aus der Schwerpunktslage ermittelten zusätzlichen Trainingsindikator $\bar{\tau}'$ – aus [8].

gewichtet werden. Dafür schlägt S. Gehlen als Adaptionsvorschrift vor:

$$p_i^{\text{neu}} = \frac{\lambda \cdot \tilde{p}_i + \kappa(1-\lambda)\, p_i^{\text{alt}}}{\kappa+1} \quad \text{mit } \lambda \text{ als Datenparameter.} \tag{17}$$

Die Zusammenfassung von Messdaten über die Vorschrift (16) hat sich gemäß [8] beim Beispiel der Produktion von α-Amylase mit Hilfe von Bacillus subtilis – einem Vorgang, der für die Hefeherstellung von Bedeutung ist – als wichtigste Verbesserung von MIAS erwiesen: Fermentationsprozesse haben große Bereiche geringer Dynamik und es finden bei dicht aufeinander folgenden Daten, die beim 50 Stunden dauernden Pilotprozess im Minutenabstand aufgenommen wurden, praktisch keine Prozessveränderungen statt. Damit führt Gleichung (16) in diesen Phasen zur Filterung der verrauschten Messsignale und die Vergrößerung von $R_{\min}$ statt zu einer Verschlechterung sogar zu einer Verbesserung der Kennfeldqualität.

Bild 11 zeigt Ergebnisse aus diesen Untersuchungen. MIAS wurde dabei mit den Daten von 4 Fermentationen zur Bildung eines Fermentationsmodells

$$\begin{bmatrix} pO_2(k) \\ pO_2(k-1) \\ CTR(k) \end{bmatrix} \rightarrow \begin{bmatrix} pO_2(k+1) \\ CTR(k+1) \end{bmatrix} \tag{18}$$

trainiert. pO_2 ist der Sauerstoffpartialdruck im Fermenter, CTR die Kohlendioxydtransferrate; kT ist der jeweilige Abtastzeitpunkt. Das Modell sagt also aus den Messwerten die im nächsten Schritt zu erwartenden Werte dieser Größen voraus. Bei 50 Stunden Prozessdauer und Abtastraten von $T = 1$ min stehen aus 4 Fermentationen $4 \times 3000 = 12\,000$ Trainingspaare zur Modellierung zur Verfügung. In Bild 11 sind die E_{bm}, die Betragsmittelwertfehler der Ausgangswerte von nicht für das Training herangezogenen Fermentationen beim größeren als für das Training verwendeten Abtastwert $T^* = 30$ min, und $R_{\min}$, der in den Rechnungen jeweils verwendete Wert für die Nutzung von Gleichung (16) dargestellt. Als Speichermaß SM wird das Verhältnis von in den Speicher als neue Punkte übernommenen zu den insgesamt beim Training verwendeten Daten bezeichnet. Ein großes $R_{\min}$ hat somit im Allgemeinen ein kleines Speichermaß zur Folge. An der Tatsache, dass sich in Bild 11 kleinere mittlere Fehler für ein großes $R_{\min}$ bzw. kleines Speichermaß ergeben, erkennt man, dass die filternde Wirkung von Glei-

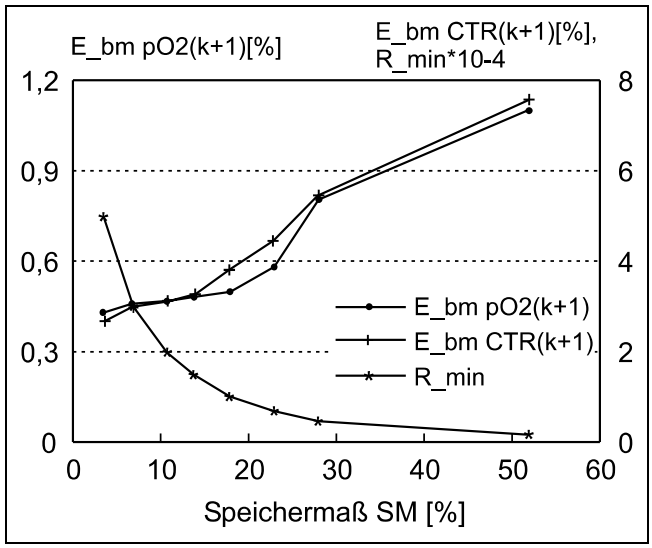


Bild 11: Betragsmittelwertfehler für die Ausgangssignale pO_2 $(k+1)$, $CTR(k+1)$ sowie der Eingangswerte als gleichen Punkt ansehende Trainingsradius $R_{\min}$ als Funktion des Speichermaßes SM; $q = 80$, $T^* = 30$ min, $\tau_{\min} = 40\%, \chi = 1/200$ in (12); $\bar{\tau}'(15)$ nicht überwacht – aus [8].

Tabelle 1: Gegenüberstellung wesentlicher Eigenschaften verschiedenartiger Ansätze zur Kennfelddarstellung ; $N =$ Anzahl der trainierten Stützstellen, $r =$ Anzahl der im Netz benutzten Neuronen.

	CMAC/AMS	MIAS	Multilayer Perceptron
Trainingsindikator	ja	ja	nein
Antwortzeit	konstant	O(N) bzw. O(log N)	O(r)
Rauschfilterung	möglich	möglich	möglich
Rechenaufwand	niedrig	mittel	mittel
Toleranz gegen Teilzerstörung	hoch	hoch	hoch
Interpolationsanpassung an lokale Stützstellendichte	nein	ja	nein
Konvergenzgeschwindigkeit	hoch	hoch	niedrig

chung (16) bei der betrachteten Fermentation in der Tat von Vorteil ist. Gleichzeitig erkennt man, dass die genaue Wahl von $R_{\min}$ unkritisch ist: Speichermaße zwischen 5% und 20% ergeben in etwa die gleiche Vorhersagegüte.

4 Wertung

Vergleicht man CMAC/AMS, MIAS und das Multilayer Perceptron mit einer verdeckten Schicht und Sigmoiden als globalen Stützfunktionen, so erhält man qualitativ das in Tabelle 1 dargestellte Ergebnis.

Prinzipiell sind alle drei Methoden zur mehrdimensionalen Kennfeldgenerierung geeignet. Allerdings ist der notwendige Trainings- und Konfigurierungsaufwand bei dem global interpolierenden Multilayer Perceptron mindestens um den Faktor zehn gegenüber den betrachteten lokal interpolierenden Speichern höher, abgesehen davon, dass dort ein zuverlässige Aussagen anzeigender Trainingsindikator fehlt, wie er beim On-line-Lernen für die dabei notwendige Überwachungs- und Sicherheitsebene wichtig ist. Deshalb sollte für das On-line-Lernen am Prozess oder im Regelkreis auf lokal verallgemeinernde Speicher zurückgegriffen werden.

Im direkten Vergleich zwischen CMAC/AMS und MIAS zeigt sich, dass die Wahl der Bewertungs- und Interpolationsparameter bei MIAS durch das sich an die Datendichte anpassende variable Interpolationsgebiet unkritischer ist als beim CMAC/AMS. Sind ausreichende Experimente zur optimalen Einstellung der freien Parameter möglich, so stellt der lokal interpolierende Speicher AMS das geeignete Werkzeug zur Prozessmodellierung dar, da auf ihn zeitdefiniert und extrem schnell zugegriffen werden kann – man ist mit der Antwortzeit nicht abhängig von einer im On-line-Betrieb sich ständig vergrößernden zu durchsuchenden Datenmenge und man hat eine etwa um den Faktor 10 kleinere Zugriffszeit als bei MIAS, gleichwertig gute Softwareimplementierungen vorausgesetzt.

Literatur

[1] An, P.C.E.; Miller, W.T. III; Parks, P.C.: Design improvements in Associative Memories for Cerebellar Model Articulation Controllers (CMAC), in: T. Kohonen et. al. (ed.) Artifical Neural Networks, Vol. 2, North Holland, 1991.

[2] Albus, J.S.: A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller. Trans. ASME. Journ. Dyn. Syst., Meas. and Control 63(3), 1975.

[3] Albus, J.S.: Theoretical and Experimental Aspects of a Cerebellar Model. Ph. D. Thesis, Univers. of Maryland, 1972.

[4] Brown, M. und Harris, C.: Neurofuzzy Adaptive Modelling and Control, 3. Auflage, Prentice Hall, 1994.

[5] Ersü, E.: On the Application of Associative Neural Network Models to Technical Control Problems. In Varju and Schnitzler, editors: Localization and Orientation in Biology and Engineering, 8. Kybernetik-Kongress, Heidelberg, FRG, March 1983. Springer 1984.

[6] Ersü, E.; Militzer, J.: Software Implementation of a Neuron-Like Associative Memory System for Control Applications. In 2nd IASTED Conference on Mini- and Micro-Computer Applications – MIMI '82, Davos, Switzerland, March 1982.

[7] Friedman, J.H.; Bentley, J.W.; Finkel, R.A.: An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Trans. Mathematical Software Vol. 3, No. 3, Sept. 1977.

[8] Gehlen, S.: Untersuchungen zur wissensbasierten und lernenden Prozessführung in der Biotechnologie. Dissertation, VDI Fortschritt-Berichte Reihe 20 (Nr. 87), VDI-Verlag, 1993.

[9] Hafner, S. (Hrsg.): Neuronale Netze in der Automatisierungstechnik. R. Oldenbourg 1994.

[10] Haykin, S.: Neural Networks, McMillan 1999.

[11] Kohonen, T.: Content-Adressable Memories, 2. Auflage, Springer 1987.

[12] Militzer, J.; Tolle, H.: Vertiefungen zu einem Teilbereiche der menschlichen Intelligenz imitierenden Regelungsansatz. In Jahrestagung der Deutschen Gesellschaft für Luft- und Raumfahrt, München, FRG, 1986.

[13] McLain, D.H.: Drawing Contours from Arbitrary Data Points, Comp. J. Vol 17, 1974.

[14] Mischo, W.S.; Kurz, A.; Tolle, H.: Einsatz eines weiterentwickelten Assoziativspeichers nach neuronalem Vorbild in einem lernenden Regelkreis. In 36. Internationales Wissenschaftliches Kolloquium, Ilmenau, FRG, 1991.

[15] Mischo, W.S.: Receptive Fields for CMAC. An Efficient Approach. In I. Aleksander and J.G. Taylor, editors, Artificial Neural Networks II. Elsevier, 1992.

[16] Mischo, W.S.: Ein neuronaler Interpolationsspeicher für die lernende Regelung: Konzeptwahl und mikroelektronischer Entwurf. Dissertation, VDI Fortschritt-Berichte Reihe 9 (Nr. 249), VDI-Verlag, 1997.

[17] Moody, J.: Fast Learning in Multi Solution Hierarchies, in: Advances in Neural Information Processing Systems, D.S. Touretzky (ed.) San Mateo, CA 1989.

[18] Renka, R.J.; Cline, A.K.: A Triangle-Based C1 Interpolation Method, Mount. J. of Math. Vol. 4, 1984.

[19] Rojas, R.: Theorie der neuronalen Netze. Springer 1996.

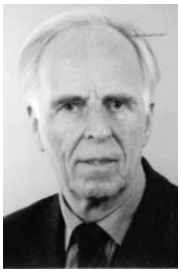
[20] Parks, P.C. und Militzer, J.: Improved Allocation of Weights for Associative Memory Storage in Learning Control Systems; Proceedings. 1. IFAC Symposium on Design Methods of Control Systems, Zürich, Sept. 1991.

[21] Schultz, J.: Idendifikation nichtlinearer dynamischer Systeme mit Künstlichen Neuronalen Netzen, Fortschritts-Berichte VDI Reihe 8 (Nr. 721), VDI-Verlag 1998.

[22] Tolle, H.; Militzer, J.; Ersü, E.: Zur Leistungsfähigkeit lokal verallgemeinernder Speicher und ihren Einsatzmöglichkeiten in lernenden Regelungen. messen – steuern – regeln msr, Bd. 32 Heft 3, 1989.

[23] Tolle, H.; Ersü, E.: Neurocontrol. Learning Control Systems Inspired by Neuronal Architectures and Human Problem Solving Strategies. In M. Thoma, editor, Lecture Notes in Control and Information Sciences No. 172. Springer, 1992.

Manuskripteingang: 21. Februar 2003.



Prof. em. Dr. rer. nat. Dipl.-Ing. Henning Tolle leitete von 1973 bis 1998 das Fachgebiet Regelsystemtheorie und Robotik am Institut für Regelungstechnik/Automatisierungstechnik der Technischen Universität Darmstadt mit den Hauptarbeitsfeldern: Regelkreissynthese, Lernende Regelkreise, Künstliche Intelligenz, Systemautonomie, Robotik und Bildverarbeitung.

Adresse: Technische Universität Darmstadt, Landgraf-Georg-Straße 4, 64283 Darmstadt,
Tel.: 06151-16-4990,
E-Mail: tolle@rt.e-technik.tu-darmstadt.de