Adaptive Resonanztheorie

Proseminar (ISI) 2005
Abt. Neuroinformatik

Moritz Gerlach
mg23@informatik.uni-ulm.de

19. April 2005

1 Einleitung

1.1 Problemstellung

Neuronale Netze bestehen aus zu Schichten zusammengefassten Neuronen, die über gewichtete Verbindungen miteinander verknüpft sind. Durch eigenständige Adaption der Gewichte sind diese Netze in der Lage, ihr Verhalten zu verändern, etwas zu lernen (*Plastizität*). Dies geschieht in der Regel dadurch, dass dem Netz immer wieder die gleichen Trainingsdaten präsentiert werden, bis sich die Gewichte optimal verteilt haben. Anschließend kann man beobachten, dass das Training eines weiteren Musters die bisher gelernten Gewichte sehr stark verändern kann und das ursprünglichen Muster verloren geht. Dieser Verlust der *Stabilität* bei gleichzeitigem Erhalt der Lernfähigkeit wird als *Stabilitäts-Plastizitäts-Dilemma* bezeichnet. [4] [5].

1.2 Einordnung

Die *Adaptive Resonanztheorie* (kurz: ART) ist ein Versuch das Stabilitäts-Plastizitäts-Dilemma zu beheben und wurde von Stephen Grossberg Mitte der siebziger Jahre des vergangenen Jahrhunderts entwickelt. Im Vergleich zu herkömmlichen neuronalen Netzen sind ART-Architekturen zum einen in der Lage, neue Muster zu erlernen, ohne die bereits bekannten zu sehr zu verändern, und zum anderen ein unüberwachtes Lernverfahren zu verwenden, welches es erlaubt, die Trainingsdaten lediglich einmal zu präsentieren (*fast learning*) [4] [5].

1.3 Überblick der Architekturen

Im Laufe der Zeit wurden verschiedene Architekturen entwickelt, die man alle unter dem Begriff ART zusammenfasst.

- Die ursprüngliche Version, ART-1, kann nur binäre Eingaben verarbeiten und verfährt wie folgt: Ein zu trainierendes Muster wird zunächst mit den bereits gelernten Mustern verglichen. Besteht Ähnlichkeit zu einem gespeicherten Muster, wird dieses modifiziert, um es dem Trainingsmuster anzupassen. Ähnelt ihm keines der gespeichertes Muster, wird eine neue Kategorie erzeugt und das Trainingsmuster abgelegt. Dabei bleiben die zuvor gespeicherten Muster unverändert erhalten. Das Stabilitäts-Plastizitäts-Dilemma ist dadurch gelöst. Eine detailliertere Beschreibung der ART-1 Architektur findet sich in Abschnitt 2.
- ART-2 ist eine Erweiterung von ART-1 und ermöglicht den Umgang mit kontinuierliche Eingaben. Der Vergleich reellwertiger Muster erfordert eine aus drei Unterschichten von Neuronen bestehende Vergleichsschicht (vgl. Abschnitt 2.1), die mit komplexen Rückkopplungsmechanismen versehen ist.

- ART-3 stellt wiederum eine Erweiterung von ART-2 dar, wobei die Gewichte nicht nur als Zahl betrachtet werden, sondern außerdem versucht wird, zeitliche und chemische Vorgänge des biologischen Vorbildes, der Synapsen, zu simulieren.
- Die ARTMAP kombiniert zwei ART Architekturen vom Typ 1 oder 2 und wendet ein überwachtes Lernverfahren an, das heißt, dass das Verhalten des Netzes in der Trainingsphase von außen überwacht und gesteuert wird.
- Fuzzy ART (siehe Abschnitt 3) stellt auch eine Erweiterung des ART-1 Netzes für reellwertige Eingabemuster dar, indem es ART-1 mit der Fuzzy-Logik verbindet. Im *fast learning* Modus haben Fuzzy ART Netze darüber hinaus die Eigenschaft, alle Trainingsmuster zu lernen, nachdem sie lediglich einmal präsentiert wurden.

Im Folgenden werden die Architekturen ART-1 und Fuzzy ART in Anlehnung an [5] näher beschrieben.

2 ART-1

2.1 Aufbau

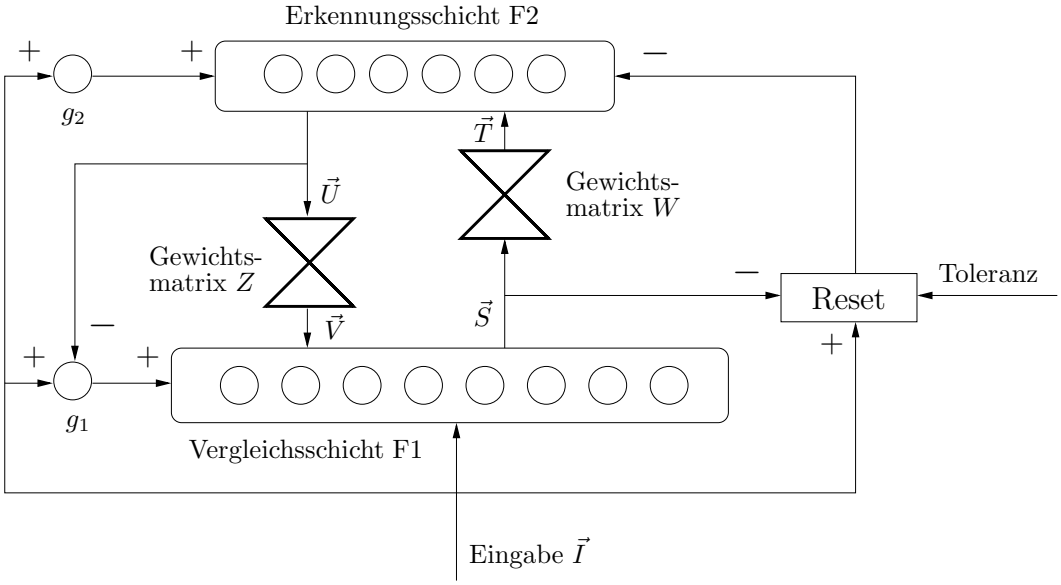

Abbildung 1: Aufbau der ART-1 Architektur

Ein ART-1 Netzwerk besteht aus zwei Schichten von Neuronen, der Vergleichsschicht (*comparison layer* mit n Neuronen) und der Erkennungsschicht (*recognition layer* mit m Neuronen), die über Gewichtsmatrizen miteinander verbunden sind (siehe Abbildung 1). Dazu kommen zwei Neuronen g_1 und g_2, die als Verstärkungsfaktoren dienen, sowie eine Reset-Komponente.
Die Vektoren $\vec{I}$, $\vec{S}$ und $\vec{V}$ haben die Dimension n, die Vektoren $\vec{T}$ und $\vec{U}$ die Dimension m, wobei sich $\vec{T}$ aus der Multiplikation des Zeilenvektors $\vec{S}$ mit der reellen $n \times m$ - Matrix W und $\vec{V}$ aus der Multiplikation des Zeilenvektors $\vec{U}$ mit der binären $m \times n$ - Matrix Z ergibt.

2.1.1 Verstärkungsfaktoren

Die Verstärkungsfaktoren und die Resetkomponente dienen als Schalter, um den Ablauf der Mustererkennung zu steuern (siehe Abschnitt 2.2). Dazu bildet g_2 das logische ODER der

Komponenten des Eingabevektors und g_1 feuert, falls $\vec{I} \neq 0$ und $\vec{U} = 0$ ist, d.h.

$$g_2 = I_1 \vee I_2 \vee \cdots \vee I_n, \quad g_1 = (I_1 \vee I_2 \vee \cdots \vee I_n) \wedge \neg(U_1 \vee U_2 \vee \cdots \vee U_m) \tag{1}$$

Damit hemmt g_2 die Erkennungsschicht, wenn $\vec{I} = 0$ anliegt (Ruhezustand) und g_1 erzeugt das unterschiedliche Verhalten der Vergleichsschicht in der Erkennungs- und Vergleichsphase (siehe Abschnitt 2.1.2)

2.1.2 Vergleichsschicht

Die Vergleichsschicht (*F1-Schicht*) erhält den binären Eingabevektor[1] $\vec{I}$, die Ausgabe des Neurons g_1 und den Vektor $\vec{V}$. Damit eine Komponente des Ausgabevektors $\vec{S}$ aktiv ist, müssen zwei der drei Eingänge den Wert 1 annehmen (*2/3-Regel*). Die i-te Komponente von $\vec{S}$ ergibt sich also aus

$$S_i = \begin{cases} 1 & \text{falls } (I_i \cdot V_i) \vee (I_i \cdot g_1) \vee (g_1 \cdot V_i) \\ 0 & \text{sonst} \end{cases} \tag{2}$$

Dies bedeutet, dass im Falle von $g_1 = 1$ $\vec{I}$ unverändert auf $\vec{S}$ abgebildet wird, da diesenfalls nach (1) $\vec{U}$ und $\vec{V}$ Nullvektoren sind. Ist $g_1 = 0$, so ist S_i genau dann 1, wenn V_i und I_i 1 sind. Die Resetkomponente liefert dann für das momentan feuernde Neuron der Erkennungsschicht eine 1, wenn sich die Vektoren $\vec{I}$ und $\vec{S}$ zu stark unterscheiden (siehe Abschnitt 2.2.2).

2.1.3 Erkennungsschicht

Die Erkennungsschicht (*F2-Schicht*) erhält die Ausgabe des Neurons g_2, den Vektor $\vec{T}$ und die vektorwertige Ausgabe der Resetkomponente. Jede Spalte von W sei mit $\vec{W_{*j}}$ bezeichnet und entspricht einem gespeicherten Muster. Die Erkennungsschicht bildet nun das Maximum über den Skalarprodukten von $\vec{S}$ mit allen Mustern $\vec{W_{*j}}$. Im Ausgabevektor $\vec{U}$ wird dann nur die dem Maximum entsprechende Komponente auf 1 gesetzt, d.h.

$$U_j = \begin{cases} 1 & \text{falls } \sum_{i=1}^{n} S_i \cdot W_{*j_i} \text{ maximal für dieses j} \\ 0 & \text{sonst} \end{cases} \tag{3}$$

2.2 Funktionsweise

Um das ART-1 Netzwerk zu initialisieren, erhalten die Komponenten der Matrix W alle den gleichen Wert, der unterhalb einer bestimmten Konstante liegen muss. Die Komponenten von Z werden alle auf 1 gesetzt und der Toleranzparameter ρ wird zwischen 0 und 1 (üblicherweise zwischen 0.7 und 0.99) gewählt. Die Toleranz bezeichnet den minimalen Wert, den das Verhältnis des skalaren Produkts von $\vec{V}$ und $\vec{I}$ zur Norm von $\vec{I}$ annehmen darf, damit $\vec{V}$ als ähnlich zu $\vec{I}$ erkannt wird.

Die Funktionsweise eines ART-1 Netzwerks, das bereits Muster nach Abschnitt 2.2.4 trainiert hat (einige Zeilen von Z entsprechen erlernten Mustern und einige Spalten von W enthalten adaptierte Gewichten), lässt sich in vier Phasen unterteilen.

2.2.1 Erkennungsphase

Zu Beginn ist $\vec{I}$ der Nullvektor und die Erkennungsschicht abgeschaltet (vgl. Abschnitt 2.1.1). Folglich sind $\vec{U}$ und $\vec{V}$ Nullvektoren. Ist nun $\vec{I}$ von 0 verschieden, feuern nach (1) g_1 und g_2 und nach (2) ist $\vec{S} = \vec{I}$. Die Erkennungsschicht bildet nun das Maximum über allen Skalarprodukten aus (3) und generiert den Vektor $\vec{U}$ derart, dass die j-te Komponente 1 ist, genau dann, wenn

[1] die Komponenten des Vektors sind Elemente aus $\{0, 1\}$

$\langle \vec{S}, \vec{W_{*j}} \rangle$ maximal ist (vgl. Abschnitt 2.1.3). Da das Skalarprodukt ein Maß für die Ähnlichkeit zweier Vektoren ist, wird jenes gespeicherte Muster gefunden, welches der Eingabe am ähnlichsten ist.

2.2.2 Vergleichsphase

Weil der Vektor $\vec{U}$ ein Einheitsvektor ist (siehe Abschnitt 2.2.1), gilt $\vec{V} = \vec{Z_{j*}}$. Da insbesondere $\vec{U}$ nicht mehr 0 ist, feuert nach (1) g_1 nicht mehr, wodurch sich nach (2) auch $\vec{S}$ ändert. Jetzt ist S_i genau dann 1, wenn V_i und I_i 1 sind (vgl. Abschnitt 2.1.2). Sind die Vektoren $\vec{V}$ und $\vec{I}$ sehr unterschiedlich, haben viele Komponenten von $\vec{S}$ den Wert 0, während die gleichen Komponenten von $\vec{I}$ den Wert 1 haben. Die Resetkomponente errechnet daraus die Ähnlichkeit und vergleicht sie mit ρ wie folgt:

$$\frac{|\vec{S}|}{|\vec{I}|} = \frac{|\vec{V} \wedge \vec{I}|}{|\vec{I}|} = \frac{|\vec{Z_{j*}} \wedge \vec{I}|}{|\vec{I}|} \geq \rho \tag{4}$$

Ist diese Ungleichung nicht erfüllt, die Toleranz also überschritten, ist das ähnlichste Muster noch zu verschieden von der Eingabe. In diesem Fall liefert die Resetkomponente eine 1 und hindert das j-te Neuron der Erkennungsschicht für die Dauer der Eingabe $\vec{I}$ daran, zu feuern.

2.2.3 Suchphase

Hat die Resetkomponente das feuernde Neuron der Erkennungsschicht gehemmt, wird $\vec{U}$ wieder zum Nullvektor, das Neuron g_1 feuert nach (1) wieder, so dass die Vergleichsschicht $\vec{I}$ erneut unverändert auf $\vec{S}$ abbildet (vgl. Abschnitt 2.1.2). Erkennungs- und Vergleichsphase wiederholen sich, wobei nun ein anderes Maximum in der Erkennungsschicht bestimmt wird. Wird dabei ein hinreichend ähnliches Muster gefunden, so beginnt das Netz mit einem Trainingszyklus und modifiziert die entsprechenden Gewichtsvektoren $\vec{W_{*j}}$ und $\vec{Z_{j*}}$. Wird kein ähnliches Muster gefunden, wird ein neuer Knoten in der Erkennungsschicht rekrutiert und dessen Gewichte in den Matrizen W und Z entsprechend gesetzt.

2.2.4 Trainingsphase

Netzwerke des ART-1 Typs kennen zwei Lernverfahren. Ähnelt das zu erlernende Muster einem bereits gespeicherten, so werden im Modus des langsamen Lernens (*slow learning*) die Gewichte in mehreren Iterationsschritten angepasst. Ist das neue Muster völlig unbekannt, arbeitet das Netz im Modus des schnellen Lernens (*fast learning*) [1] und die Komponenten der j-ten Spalte der reellen Matrix W ergeben sich aus normalisierten Einträgen des Vektors $\vec{S}$ wie folgt:

$$\vec{W_{ij}} = \frac{L \cdot S_i}{L - 1 + \sum_{k=1}^{n} S_k} \tag{5}$$

mit einer Konstanten $L > 1$ (typischerweise $L = 2$). Der binäre Vektor $\vec{Z_{j*}}$ übernimmt die Einträge von $\vec{S}$ unverändert [5].

3 Fuzzy ART

3.1 Fuzzy Mengen

Die Fuzzy Logik wurde entwickelt, um unscharfe Bereiche auf präzise Signale abzubilden. Dabei sieht man sich mit dem Problem konfrontiert, ungenaue Ausdrücke in exakte Maschinenanweisungen übersetzen zu müssen. Für diese Kategorisierung wurde das Konzept der Fuzzy Mengen entwickelt, welches ermöglicht, dass ein Element auch nur zu einem Teil in einer Menge enthalten sein kann (z.B. kann eine Temperatur von 15°C zu einem Teil in der Menge der *angenehmen* wie auch der *frischen* Temperatur liegen) [2].

Formal gibt es zu jeder Menge A eine Funktion μ_A, die jedes Element auf ihren Zugehörigkeitsgrad, d.h. eine reelle Zahl aus dem Intervall $[0, 1]$, abbildet. Die komplementäre Zugehörigkeitsfunktion ist gegeben durch $1-\mu_A(x)$. Für den Durchschitt (logisches UND) und die Vereinigung (logisches ODER) zweier Fuzzy Mengen A und B ist die Zugehörigkeitsfunktion jeweils wie folgt definiert:

$$\mu_{A\cap B}(x) = \min\{\mu_A(x)\,, \mu_B(x)\}$$
$$\mu_{A\cup B}(x) = \max\{\mu_A(x)\,, \mu_B(x)\}$$

Ferner ist nach [3] die Kardinalität einer Fuzzy Menge als $|A| = \sum_{x\in A} \mu_A(x)$ definiert.

3.2 Aufbau

Bei einem Fuzzy ART Netz begreift man die Vektoren als Fuzzy Mengen und deren Einträge als entsprechende Werte der Zugehörigkeitsfunktion. Dementsprechend besteht der Eingabevektor $\vec{I}$ aus reellen Einträgen, normiert auf das Intervall $[0, 1]$, die außerdem häufig komplementär kodiert sind. Letzteres bedeutet, dass der Vektor $\vec{I} = (a_1, \ldots, a_n)$ um die komplementären Einträge erweitert wird, so dass man $\vec{I} = (a_1, \ldots, a_n, 1-a_1, \ldots, 1-a_n)$ erhält. Damit ist stets die Kardinalität $|I| = n$.
Jedes Neuron der Erkennungsschicht steht wie bei der ART-1 Architektur für ein gespeichertes Muster (bzw. einen freien Speicherplatz), jedoch existiert nur je ein reellwertiger normierter Gewichtsvektor $\vec{W_j}$.

3.3 Funktionsweise

Zu Beginn werden die Gewichtsvektoren mit 1 initialisiert, d.h. $W_{ij} = 1\,\forall j, i$. Die Auswahl des ähnlichsten Musters aus $\vec{W_j}$ bei Eingabe $\vec{I}$ ergibt sich analog zu (3) aus

$$T_j = \frac{|I \cap W_j|}{\alpha + |W_j|} \tag{6}$$

Durch den Parameter $\alpha > 0$ wird eine Division durch Null verhindert. Um aber T_j nicht zu stark zu verfälschen, wird $\alpha \to 0$ angenommen. Sei J das Neuron der Erkennungsschicht, dessen T_J maximal[2] ist. Analog zu (4) wird nun die tatsächliche Ähnlichkeit von $\vec{I}$ und $\vec{W_J}$ geprüft, indem die Ungleichung

$$\frac{|I \cap W_J|}{|I|} \geq \rho \tag{7}$$

erfüllt sein muss. Ist das nicht der Fall, wird erneut das Maximum aller T_j bestimmt, wobei die Resetkomponente wieder dafür sorgt, dass das Neuron J für die Dauer der Eingabe $\vec{I}$ nicht mehr ausgewählt wird. Existiert kein ähnliches Muster, fällt das Maximum schließlich auf einen Vektor $\vec{W_J} : W_{Ji} = 1\,\forall i$.
Ist (7) erfüllt, dann werden die Gewichte komponentenweise wie folgt angepasst:

$$W_J(t+1) = \eta \cdot (I \cap W_J(t)) + (1-\eta) \cdot W_J(t),\ \eta \in [0, 1] \tag{8}$$

Im Modus des schnellen Lernens wird die Lernrate $\eta = 1$ gewählt, wodurch der zweite Teil des Terms entfällt und es daher genügt, nur eine Iteration auszuführen. Eine weitere Möglichkeit ist die sog. *fast-commit slow-recode* Option, für die man zunächst $\eta = 1$ wählt und anschließend weitere Iterationen mit $\eta < 1$ durchführt, wobei der erste Teil des Terms unverändert bleibt und daher $\vec{I}$ nicht mehr anliegen muss.
Die Einträge der Vektoren $\vec{W_j}$ können sich während des Training nur verringern oder gleichbleiben, so dass sie gegen eine Grenze konvergieren müssen.

[2] J ist minimal zu wählen, falls das Maximum $\max_j T_j$ für verschiedene j angenommen wird. Dadurch ist J eindeutig bestimmt.

Unter Verwendung des schnellen Lernens normierter und komplementär kodierter Eingabevektoren sowie mit $\alpha \to 0$, ist das Verfahren stabil, d.h. es genügt, jedes Muster einmal zu präsentieren.

3.4 Geometrische Interpretation der Adaption

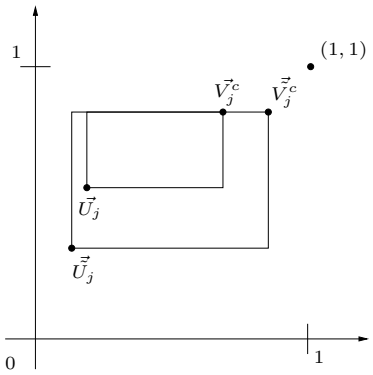

Abbildung 2: Adaption eines zweidimensionalen Musters.

Betrachten wir vereinfachend zweidimensionale, durch die komplementäre Kodierung zu vierdimensionalen Vektoren vergrößerte Eingaben, so lässt sich die Arbeitsweise von Fuzzy ART mit Rechtecken veranschaulichen: Sei $\vec{I} = (\vec{A}, \vec{A}^c)$ und der Gewichtsvektor $\vec{W}_j = (\vec{U}_j, \vec{V}_j)$ durch zweidimensionalen Vektoren $\vec{A}$, $\vec{U}_j$ und $\vec{V}_j$ gegeben. Betrachten wir nun das Rechteck, welches von $\vec{U}_j$ (untere linke Ecke) und $\vec{V}_j^c$ (obere rechte Ecke) aufgespannt wird. Wird ein zu lernendes Muster in einer neuen Klasse abgelegt, gilt $\vec{U}_j = \vec{A}$ und $\vec{V}_j = \vec{A}^c$. Oben definiertes Rechteck besteht diesenfalls also aus nur einem Punkt. Soll ein Muster $\vec{W}_j$ gemäß einer neuen Eingabe $\vec{I}$ angepasst werden, so wird $\vec{U}_j$ zu $\tilde{\vec{U}}_j = \min\{\vec{U}_j, \vec{A}\}$ und $\vec{V}_j$ zu $\tilde{\vec{V}}_j = \min\{\vec{V}_j, \vec{A}^c\} = \max\{\vec{V}_j^c, \vec{A}\}^c$. In Abbildung 2 sieht man, dass sich die untere linke Ecke $\tilde{\vec{U}}_j$ in Richtung des Nullpunkts und die obere rechte Ecke $\tilde{\vec{V}}_j^c$ in Richtung der Koordinate $(1, 1)$ bewegt. Beschränkt wird diese Bewegung durch den Toleranzparameter ρ, denn aus (7) folgt, dass stets $|W_j| \geq 2 \cdot \rho$ gelten muss, damit der Vektor $\vec{W}_j$ als ähnlich zur Eingabe $\vec{I}$ erkannt wird [4] [5].

4 Zusammenfassung

Neuronale Netzwerke des Typs ART-1 und Fuzzy ART lösen erfolgreich das Stabilitäts-Plastizitäts-Dilemma. Jede Eingabe wird zunächst mit den bereits gespeicherten Mustern verglichen und nur, wenn ein hinreichend ähnliches gefunden wird, modifiziert. Andernfalls wird das zu speichernde Muster in einer separaten Klasse abgelegt und die alten Muster bleiben unverändert erhalten. Obwohl ART-1 Netzwerke auf binäre Eingabevektoren beschränkt sind, bildet die Fuzzy Logik die Grundlage für eine elegante Erweiterung zu Fuzzy ART. Hiermit können reellwertige Muster klassifiziert werden, ohne dass die Struktur des Netzes grundlegend verändert werden muss.

Literatur

[1] S. Meretz, *Lernkonzepte in Theorien "Neuronaler Netze"*, http://www.opentheory.org/neuro_netze_kap_4_4/text.phtml vom 13.02.2005

[2] G. Reif, *Moderne Aspekte der Wissensverarbeitung*, Diplomarbeit an der Technischen Universität Graz, Kapitel 7 *Fuzzy Logik*, http://www.iicm.edu/greif/node9.html vom 11.02.2005

[3] B. T. Luke, *Fuzzy Sets and Fuzzy Logic*, http://members.aol.com/btluke/fuzzy01.htm vom 13.02.2005

[4] G. Turi, *Fuzzy - ART (MAP)*, http://sphinx.rbi.informatik.uni-frankfurt.de/~paetz/NFS2.pdf vom 13.02.2005

[5] A. Zell, *Simulation Neuronaler Netze*, Addison-Wesley 1996, Kapitel 22.1 und 22.6