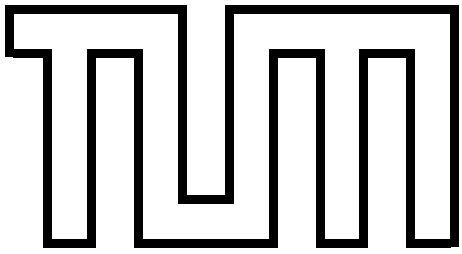

Technische Universität München

Fakultät für Informatik

Forschungs- und Lehreinheit Informatik VI

Unsupervised Learning

Seminar Kognitive Robotik (SS12)

Sebastian Fröhlich

Betreuer: Florian Röhrbein

Leitung: Prof. Alois Knoll

Abgabetermin: 21. Juli 2012

Inhaltsverzeichnis

Zusammenfassung

Das Thema dieses Proseminars ist das unüberwachte Lernen. Es handelt sich hierbei um maschinelles Lernen, welches ohne vorherige Festlegung von Kategorien Objekte klassifiziert. Dies geschieht ohne Lehrer. Um diese Art des Lernens darzustellen, wurde die von G. Carpenter und S. Grossberg entwickelte Adaptive Resonanz Theorie, im Detail der ART 1 verwendet.

1 Einführung

Häufig existiert eine Menge von Eingaben die klassifiziert werden soll. Wenn bereits mögliche Gruppierungen bzw. Muster bekannt sind, ist eine Umsetzung mit überwachtem Lernen denkbar. Dieses ordnet gegebene Muster (z.B. durch Adjustierung der Gewichte) den derzeitigen Gruppierungen zu. Des öfteren tritt aber der Fall auf, dass es sich um eine Menge handelt, deren Muster nicht bekannt ist. An dieser Stelle spielt unüberwachtes Lernen eine wichtige Rolle. Die Klassifizierung der Muster findet hier nicht aufgrund möglicher Ausgaben statt, stattdessen wird von einem unüberwachtem Netz gefordert, es solle mögliche Klassen aufgrund der Ähnlichkeit unter den Eingabemustern selbständig formen und zuweisen.

1.1 Competitive Learning

Das kompetitive Lernen oder Competitive Lerning ist ein Verfahren, welches ohne Vorgabe eines bestimmten Outputs funktioniert. Allgemein ermittelt es vielmehr aufgrund bestimmter Gegebenheiten einen 'Gewinner' und ordnet Muster dementsprechend zu. Während des Trainings werden hierzu die Gewichte des Eingangs zu allen Ausgangseinheiten verglichen. Anschließend wählt das Netz durch Wettbewerb die Ausgabe, die die höchste Gewichtszuordnung besitzt. Als Folge dessen werden alle Gewichte die zur Gewinnereinheit führen entsprechend zum Eingang verändert. Dieses Prinzip nennt sich *'The winner takes it all'*. Diese Art der Anpassung hat allerdings den Nachteil, dass ein einzelner Ausgang im Extremfall alle Eingänge für sich beanspruchen kann, während andere keine oder fast keine erlangen. So ist keine sinnvolle Klassifizierung mehr möglich.

1.2 Das Stabilitäts-Plastizitäts-Dilemma

Grundsätzlich sollte es jedem Netz möglich sein, neue Muster zu erlernen. Dieser Umstand fällt unter den Begriff der Plastizität. Plastizität bezeichnet im Allgemeinen die Möglichkeit neuronaler Netze, sich bei bestimmten Gegebenheiten zu verändern. Bei vielen Verfahren existiert allerdings das Problem, dass bereits Gelerntes wieder vergessen wird. Diese Netze gelten als nicht stabil. Um das Stabilitäts-Plastizitäts-Dilemma zu lösen, muss in einem neuronalem Netz demnach,

- sowohl die Möglichkeit des Erlernens neuer Muster,
- als auch das Bestehen bleiben vorhandener Muster

gegeben sein. Das nachfolgend aufgeführte Verfahren der Adaptiven Resonanz Theorie zeigt eine Art dieses Dilemma unüberwacht zu lösen.

2 Die Adaptive Resonanztheorie

Die Adaptive Resonanz Theorie (kurz ART) ist kein einzelnes Modell, sondern ein Überbegriff vieler Lernverfahren. Nachfolgend die wichtigsten Vertreter:

- *ART-1:* ursprüngliche Version, nur für binäre Eingaben geeignet [5]
- *ART-2:* Erweiterung von ART-1 für kontinuierliche Eingabevektoren [5]
- *ART-2A:* Vereinfachung von ART-2 für schnellere Konvergenz des Netzes [5]
- *Fuzzy Art:* Kombination von Fuzzy Logic und ART [5]

Die Adaptive Resonanztheorie behandelt das Stabilitäts-Plastizitäts-Dilemma biologisch plausibel. Während des Einlesens neuer Muster, besitzen die Algorithmen die Möglichkeit bereits Gelerntes heranzuziehen und auf Ähnlichkeit zu überprüfen. Bei einer Übereinstimmung wird das vorhandene Muster im Prinzip nicht gelöscht sondern verbessert. So bleibt die Stabilität gewahrt und das neue Muster wird gleichzeitig in das bereits Vorhandene integriert.

3 ART-1

In allen folgenden Erläuterungen und Formeln gilt sofern nicht anders beschrieben:

$$i \in \{0, \ldots, m\}$$

$$j \in \{0, \ldots, n\}$$

Wobei m die Anzahl der Bits des Eingabevektors und n die Anzahl der Kategorien darstellt.

3.1 Allgemeine Vorgehensweise

Die nachstehende Grafik erläutert die allgemeine Vorgehensweise des ART-1:

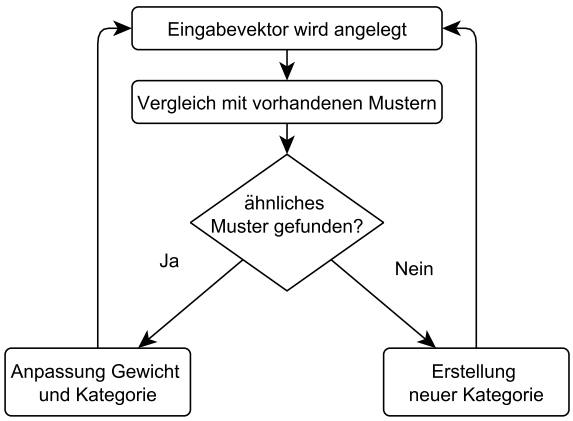

Abbildung 1: Allgemeines Vorgehen ART-1

Ein Muster gilt als ähnlich, sobald es, bis auf eine gewisse Toleranz, mit dem neuen Muster übereinstimmt. Alle Muster, die nicht betroffen sind, werden nicht modifiziert.

3.2 Die Architektur

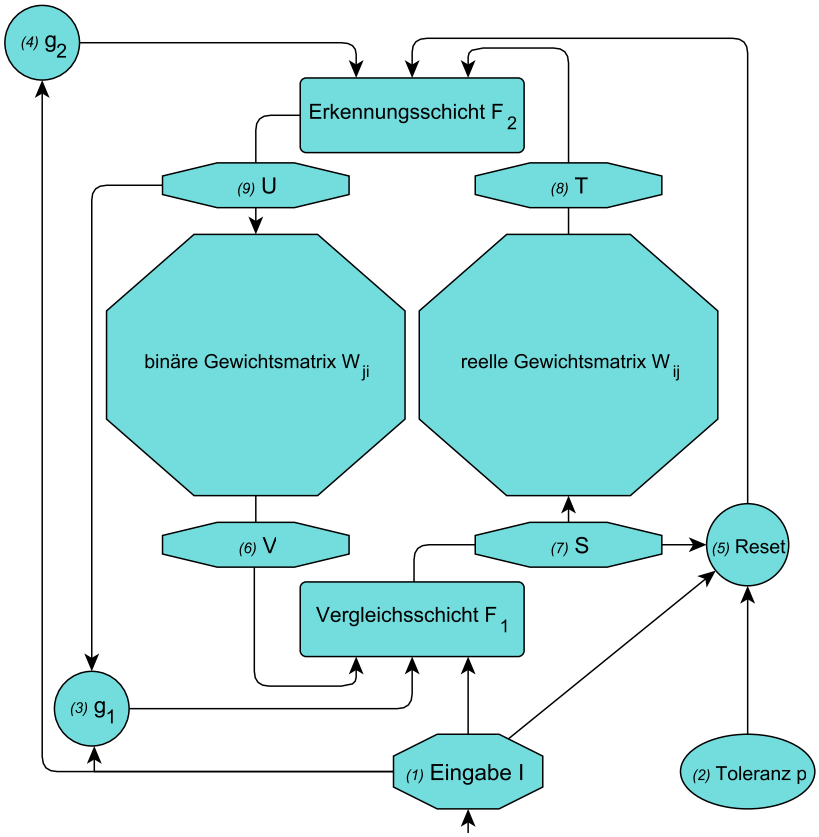

Abbildung 2: Architektur ART-1

3.2.1 Binärer Eingabevektor $\vec{I}$

Der ART-1 wurde entwickelt um binäre Eingabemuster zu vergleichen. Aus diesem Grund gilt für den Eingabevektor $\vec{I}$:

$$\vec{I} = (I_1, I_2, ..., I_m) \text{ für } I_i \in \{0,1\} \tag{1}$$

3.2.2 Toleranzparameter p

Allgemein gilt:

$$0 < p \leq 1 \tag{2}$$

Die Größe des Toleranzparameters hat hohen Einfluss auf die Klassifizierung. Bei Wahl eines kleinen Toleranzparameters findet die Klassifizierung schneller statt. Dies führt zu groben Klassen. Je höher p gewählt wird, desto genauer wird klassifiziert. P sollte normalerweise trotzdem nicht den Wert 1 erhalten, da sonst nur 100% übereinstimmende Muster als ähnlich gelten.

3.2.3 Verstärkungsfaktoren g_1 und g_2

Die Namensgebung der beiden Verstärkungsfaktoren g_1 und g_2 ist etwas irreführend, da sie nicht als Verstärkung, sondern als Schalter für die Vergleichs- bzw. Erkennungsschicht fungieren.

$$g_1 = \begin{cases} 1, & (I_1, \ldots, I_m) \neq (0, \ldots, 0) \wedge (U_1, \ldots, U_m) = (0, \ldots, 0) \\ 0, & \text{sonst} \end{cases} \tag{3}$$

Der Verstärkungsfaktor g_1 ermöglicht der $F_2 - Schicht$ einen Vergleich zwischen $\vec{V}$ und $\vec{I}$, sobald es eine Eingabe gibt und gleichzeitig von der Erkennungsschicht ein Muster erkannt wurde.

$$g_2 = \begin{cases} 1, & (I_1, \ldots, I_m) \neq (0, \ldots, 0) \\ 0, & \text{sonst} \end{cases} \tag{4}$$

Der Verstärkungsfaktor g_2 hindert die Erkennungsschicht, bei fehlender Eingabe, ein Muster auszuwählen.

3.2.4 Reset

Der Reset hat die Funktion eine Schaltung der Erkennungsschicht zu verhindern, vorausgesetzt, der Unterschied zwischen dem Ausgabevektor der Vergleichsschicht $\vec{S}$ und dem Eingangsvektor $\vec{I}$ ist größer als der Toleranzparameter p zulässt.

Der Unterschied wird durch einen Vergleich der Anzahl der Einsen gemessen.

$$Reset = \begin{cases} 1, & \frac{\sum_i^m S_i}{\sum_i^m I_i} < p, mit\, 0 < p \leq 1 \\ 0, & \text{sonst} \end{cases} \tag{5}$$

3.2.5 Die Vergleichsschicht F_1

Die Vergleichsschicht wird zum Vergleich einzelner Neuronen, in diesem Fall einzelner Bits verwendet.

Eingangsvektor $\vec{V}$ der Vergleichsschicht

$$V_i = \sum_{j=1}^{n} U_j * W_{ji} \tag{6}$$

Der Eingangsvektor ist hierbei eine Repräsentation des Musters der binären Gewichtsmatrix W_{ji}, welches von der Erkennungsschicht als am Ähnlichsten deklariert wurde.

Ausgabevektor $\vec{S}$ der Vergleichsschicht

Jedes Neuron der Vergleichsschicht berechnet sich aus allen Komponenten

- I_i des Eingabevektors $\vec{I}$ (1),
- V_i des übernommenen Musters der Top-Down Matrix W_{ji} (6)
- und dem Verstärkungsfaktor g_1(3)

Für die Ausgabe $\vec{S}$ von F_1 gilt:

$$S_i = \begin{cases} 1, & \text{falls } (I_i * V_i) \vee (g_1 * V_i) \vee (I_i * g_i) = 1 \\ 0, & \text{sonst} \end{cases} \tag{7}$$

Durch diese Regel ist gegeben, dass mind. 2 der 3 Eingangs-Komponenten den Wert 1 besitzen müssen, um entsprechend eine 1 als Ausgabe S_i zu erhalten (*2/3-Regel*). Dies bedeutet, dass $\vec{S}$ entweder eine Kopie des Eingabevektors $\vec{I}$ im Falle $g_1 = 1$ ist oder das logische *'und'* von $\vec{S}$ und $\vec{I}$ darstellt. Dieses *'und'* stellt den Vergleich zwischen der Eingabe und einem vorhanden Muster dar.

Zu Beginn der Ausführung liegen an der Vergleichsschicht $g_1 = 1$ und $\forall i \in \{1, \ldots, m\}$ $V_i = 0$ an, d.h. es tritt der erste Fall mit $S_i = I_i$ auf.

3.2.6 Die Erkennungsschicht F_2

Die Erkennungsschicht wird, im Gegensatz zur Vergleichsschicht, zur Erkennung einer Kategorie verwendet. Die Anzahl der Kategorien muss nicht gleich der Anzahl der Bits des Musters sein. Aus diesem Grund kann sich die Anzahl der Neuronen der Erkennungsschicht von denen der Vergleichsschicht unterscheiden.

Eingang $\vec{T}$ der Erkennungsschicht

Der Eingangsvektor ist die gewichtete Summe des Musters $\vec{S}$ und der reellen Gewichtsmatrix W_{ij} und stellt somit ein Maß für die Ähnlichkeit eines Musters mit einer Kategorie dar.

$$T_j = \sum_{i=1}^{m} S_i * W_{ij} \tag{8}$$

Ausgabe $\vec{U}$ der Erkennungsschicht

Die Ausgabe der Erkennungsschicht soll wie zuvor beschrieben eine passende Kategorie ermitteln. Zu diesem Zweck wird die Kategorie gewählt, die die größte Ähnlichkeit besitzt. Es existiert keine einheitliche Regelung zur Feststellung der größten Ähnlichkeit. Eine gebräuchliche Möglichkeit ist die Ausgabe der maximalen Stelle des Eingangsvektors $\vec{U}$. Diese Möglichkeit wird auch bei allen weiteren Berechnungen dieser Arbeit verwendet.

Jedes Neuron der Erkennungsschicht berechnet sich folglich aus den Komponenten T_j des Eingangsvektors $\vec{T}$ (8). Die Ausgabe kann allerdings durch

- den Verstärkungsfaktor g_2(4)
- und den *Reset* (5)

verhindert werden.

Die Berechnung selbst erfolgt durch:

$$U_j = \begin{cases} 1, & \text{falls } T_j = max(\sum_{i=1}^{m} S_i * W_{ij}) \wedge g_2 = 1 \wedge Reset = 0 \\ 0, & \text{sonst} \end{cases} \tag{9}$$

3.3 Vorgehensweise

Die Funktionsweise des ART-1 besteht aus folgenden Phasen:

1. Initialisierung
2. Erkennungsphase (recognition)
3. Vergleichsphase (comparison)
4. Suchphase (search)
5. Adaption der Gewichte (training)

Nachfolgend werden die einzelnen Phasen genauer erklärt. Ein grafisches Abbild der Vorgehensweise befindet sich in Anhang *A*.

3.3.1 Initialisierung

Die Gewichte der Buttom-Up Matrix W_{ij} werden für alle Elemente zufällig unter der Bedingung

$$w_{ij} \leq \frac{L}{L-1+m}, \text{ mit } m = |I| \, und \, L \geq 2 \tag{10}$$

gesetzt.

L ist eine Konstante und wird gewöhnlich als L = 2 gewählt. Es ist hierbei wichtig, keine zu großen w_{ij} zu wählen, da sonst die Möglichkeit besteht, dass alle Eingabevektoren auf das selbe Neuron J abgebildet werden.

Das Setzen der Gewichte der Top-Down Matrix W_{ji} erfolgt für alle Elemente durch:

$$w_{ji} = 1 \tag{11}$$

Der Toleranzparameter p wird auf einen gewünschten Wert gesetzt, für den die Bedingungen *(3.2.2)* gelten.

3.3.2 Erkennungsphase (recognition)

Durch das Setzen des Nullvektors als Eingangsvektor $\vec{I}$, stellt sich g_2 auf 0 *(4)* ein. Da F_2 nun keine Ausgabe erzeugen kann, gibt sie für $\vec{U}$ den Nullvektor zurück *(9)*. Dementsprechend ist $\vec{V}$ ebenfalls 0 *(6)*.

Anschließend kann ein vom Nullvektor abweichender Eingabevektor angelegt werden. Als Folge ändern sich g_1 und g_2 auf 1 *(3 / 4)*. Durch das aktuell gegebene $\vec{V} = 0$ sowie $g_1 = 1$

ist $\vec{S}$ ein Duplikat von $\vec{I}$. Bei Einlesen des nächsten Eingabevektors muss dies nicht mehr so sein. Es sei denn es kam zu einem Reset. Dies wird im nächsten Punkt genauer erläutert. Anschließend wird für jedes Neuron S_i sein Skalarprodukt mit W_{ij} gebildet *(8)* um $\vec{T}$ zu berechnen. Da $\vec{T}$ als Maß für die Ähnlichkeit der Muster genutzt wird, wird nun das Neuron mit der höchsten Ähnlichkeit, also das Maximum von T ermittelt und anschließend als $\vec{U}$ weitergegeben *(9)*. $\vec{U}$ übermittelt folglich nur 1 Bit mit den Informationen, welches Neuron die höchste Ähnlichkeit besitzt, während alle anderen 0 sind.

3.3.3 Vergleichsphase (comparison)

Durch das Signal, welches $\vec{U}$ übermittelt erhält man $\vec{V}$ *(6)*. $\vec{V}$ stellt hierbei die Zeile der Binärmatrix W_{ji} dar, welche durch das Gewinnerneutron in $\vec{U}$ definiert wurde. Da sich soeben die Bedingung $\vec{U} = 0$ *(3)* für g_1, durch die Berechnung von $\vec{U}$, verändert hat, gilt ab sofort $g_1 = 0$.
Als Resultat der vorherigen Ergebnisse, folgt $\vec{S}$ als logisches *'und'* zwischen $\vec{V}$ und $\vec{I}$ *(7)*. Dies bedeutet, dass alle Komponenten der Eingabe deaktiviert werden, die nicht mit dem Muster übereinstimmen.

Dieser Schritt ist wichtig, denn durch anschließendes Zählen der Einsen bzw. Nullen, kann ein Vergleich zwischem dem Eingabevektor und der Übereinstimmung $\vec{S}$ stattfinden.
Unterscheidet sich der Quotient der beiden Summen um mehr als der Toleranzparameter p *(2)* zulässt, wird der Reset ausgelöst *(5)* und es findet kein weiterer Vergleich mit diesem Muster mehr statt. An dieser Stelle wird sichergestellt, dass sich keine falsche Klassifizierung einstellen kann. Der Algorithmus tritt hierbei in die Suchphase über. Wird der Reset nicht ausgelöst geht er zur 5. Phase, der Adaption der Gewichte, über.

3.3.4 Suchphase (search)

Zu Beginn der Suchphase wird die abweichende Kategorie, also das soeben nicht im Toleranzbereich liegende Muster deaktiviert. Wenn nun mindestens noch eine Kategorie aktiv ist kann $\vec{U} = 0$, mit dem Ergebnis $g_1 = 1$, gesetzt werden. Dies geschieht um $\vec{S}$ wieder in den Ursprungszustand, also der Eingabe $\vec{I}$ zurückzusetzen um erneut mit der Berechnung von T_i zu beginnen.
Dieser Prozess wird solange ausgeführt, bis entweder ...

- ... keine aktive Kategorie mehr vorhanden ist.
  In diesem Fall wird eine neue Kategorie mit entsprechenden Gewichten erstellt. Anschließend kann ein neues Muster als Eingabe $\vec{I}$ angelegt werden um wieder von vorne zu beginnen.

  oder

- ... ein im Toleranzbereich liegendes Muster gefunden wird.
  In diesem Fall geht der Prozess zur 5. Phase der Adaption der Gewichte über.

3.3.5 Adaption der Gewichte (training)

In ART-Netzwerken wird zwischen 2 Arten der Adaption unterschieden:

- Langsames Training (slow training bzw. slow learning)
  Der Eingabevektor wird hier nur kurz angelegt, somit können die Gewichte sich nicht annähern, sondern werden statisch verteilt.

  und

- Schnelles Training (fast training bzw. fast learning)
  Der Eingabevektor wird hier dementsprechend lange angelegt, bis alle Gewichte einen stabilen Wert erreichen. Die Berechnung erfolgt durch

$$w_{iJ} = \frac{L * s_i}{L - 1 + \sum_k s_k} \tag{12}$$

$$w_{Ji} = s_i \tag{13}$$

  wobei L derselbe Parameter L ist, mit welchem zu Beginn die Gewichte initialisiert wurden.

3.4 Anwendungsbeispiel

Im folgenden Anwendungsbeispiel sollen nacheinander die Eingabevektoren $I_1 = (101)$ und $I_2 = (011)$ angelegt werden um die Vorgehensweise zu verdeutlichen. Hierbei kann gut beobachtet werden, dass die Musterklassifizierung ohne vorherige Festlegung der Muster erfolgt. Das Verfahren arbeitet dementsprechend ohne Lehrer, also unsupervised.

Phase 1: Initialisierung

1.1 $W_{ij} \leq \frac{L}{L-1+m}$, *mit* $m = |I| = 3$ *und* $L = 2$

1.2 $W_{ji} = \quad 1$

Phase 2: Erkennungsphase

1.3 Lege Nullvektor $\vec{I} = (000)$ an

1.4 $g_2 = \quad 0 \quad$, da $(I_1, \ldots, I_m) = (0, \ldots, 0)$

1.5 $U = (000) \quad$, da $g_2 = 0$

1.6 $V = (000) \quad$, da $U = (000)$

2.1 Lege neuen Eingabevektor $\vec{I} = (101)$ an

2.2 $g_1 = \quad 1 \quad$, da $I \neq \vec{0} \wedge \vec{U} = \vec{0}$

2.3 $S_i = \quad (I_i * V_i) \vee (g_1 * V_i) \vee (I_i * g_1)$

$S_1 = \quad 1 \quad = \quad (1 * 0) \vee (1 * 0) \vee (1 * 1)$

$S_2 = \quad 0 \quad = \quad (0 * 0) \vee (1 * 0) \vee (0 * 1)$

$S_3 = \quad 1 \quad = \quad (1 * 0) \vee (1 * 0) \vee (1 * 1)$

2.4 $R = \quad 0 \quad$, da $\frac{\sum_i^m S_i}{\sum_i^m I_i} = \frac{2}{2} = 1 > 0,7$

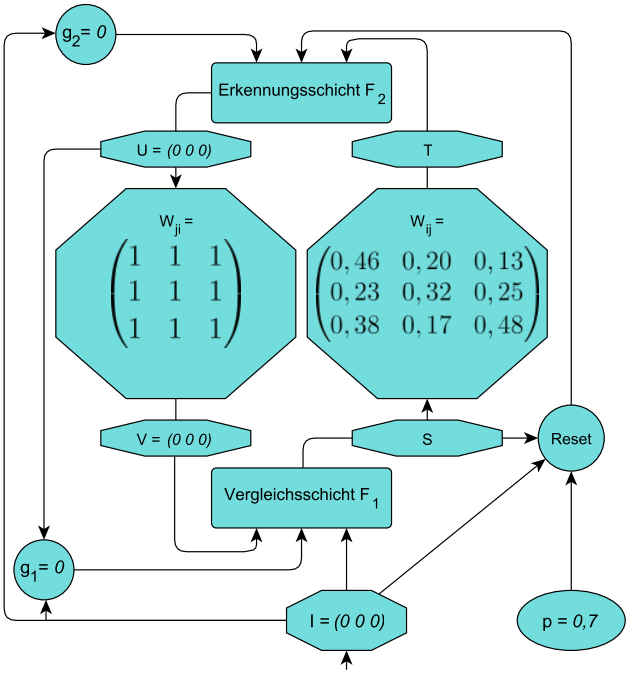

Abbildung 3: Schritt *1.1-1.6*

$$
\begin{aligned}
2.5 \quad T_j &= S_1W_{1j} + S_2W_{2j} + S_3W_{3j} \\
T_1 = 0,84 &= 1*0,46 + 0*0,23 + 1*0,38 \\
T_2 = 0,37 &= 1*0,20 + 0*0,32 + 1*0,17 \\
T_3 = 0,61 &= 1*0,13 + 0*0,25 + 1*0,48 \\
2.6 \quad U &= (100) \quad \text{, da } T_1 \text{ max}
\end{aligned}
$$

Phase 3: Vergleichsphase

$$
\begin{aligned}
2.7 \quad V_i &= = U_1W_{1i} + U_2W_{2i} + U_3W_{3i} \\
V_1 = 1 &= 1*1 + 0*1 + 0*1 \\
V_2 = 1 &= 1*1 + 0*1 + 0*1 \\
V_3 = 1 &= 1*1 + 0*1 + 0*1
\end{aligned}
$$

$$
\begin{aligned}
2.8 \quad g_1 &= 0 \quad \text{, da } U \neq \vec{0} \\
2.9 \quad S_i &= (I_i * V_i) \vee (g_1 * V_i) \vee (I_i * g_1) \\
S_1 = 1 &= (1*1) \vee (1*1) \vee (1*1) \\
S_2 = 0 &= (0*1) \vee (1*1) \vee (0*1) \\
S_3 = 1 &= (1*1) \vee (1*1) \vee (1*1) \\
2.10 \quad R &= 0 \quad \text{, da } \frac{\sum_i^m S_i}{\sum_i^m I_i} = \frac{2}{2} = 1 > 0,7
\end{aligned}
$$

Phase 5: Adaption der Gewichte

2.11 Anpassung der Gewichte

$$
\begin{aligned}
W_{ij} &= \frac{L*S_i}{L-1+\sum_{k=1}^{j} S_k} \\
W_{11} = \tfrac{2}{3} &= \tfrac{2*1}{2-1+2} \\
W_{21} = 0 &= \tfrac{2*0}{2-1+2} \\
W_{31} = \tfrac{2}{3} &= \tfrac{2*1}{2-1+2} \\
W_{Ji} &= s_i \\
W_{11} = 1 &= s_1 \\
W_{12} = 0 &= s_2 \\
W_{13} = 1 &= s_3
\end{aligned}
$$

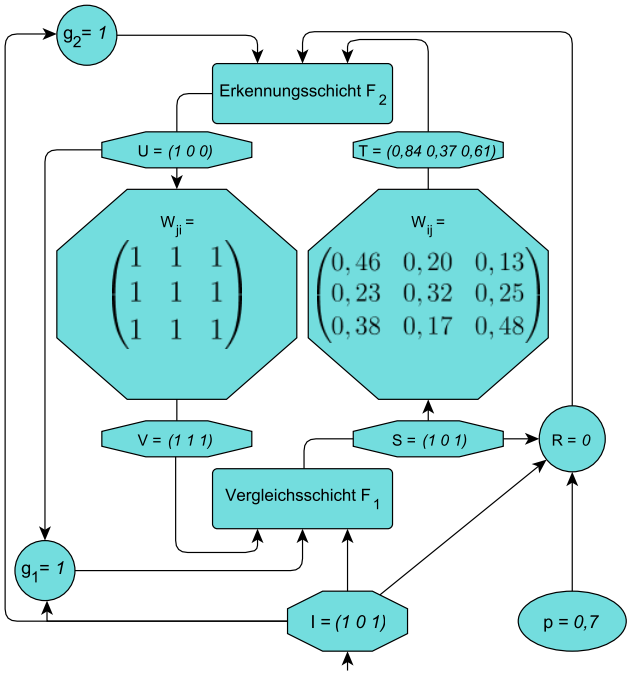

Abbildung 4: Schritt *2.1-2.7*

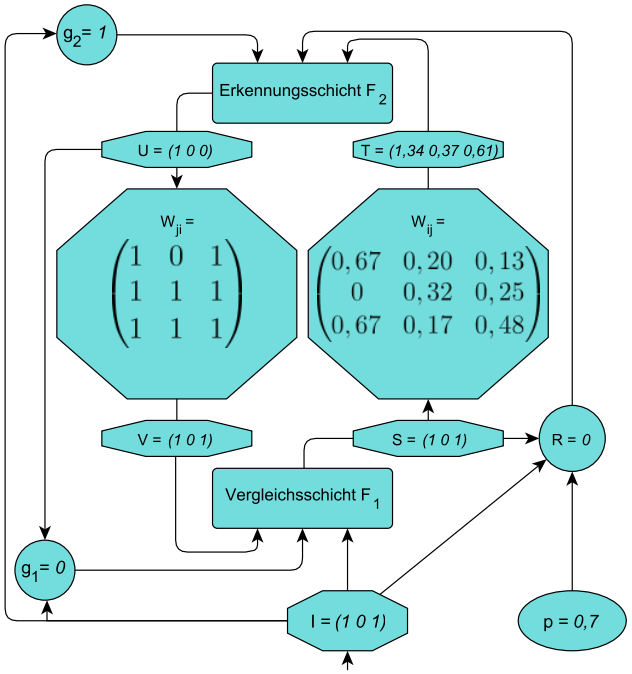

Abbildung 5: *Schritt 2.8-2.14*

2.12 $T_j = \quad S_1W_{1j} + S_2W_{2j} + S_3W_{3j}$

$T_1 = 1,34 = 1*0,67 + 0*0,00 + 1*0,67$

$T_2 = 0,37 = 1*0,20 + 0*0,32 + 1*0,17$

$T_3 = 0,61 = 1*0,13 + 0*0,25 + 1*0,48$

2.13 $U = (100)$, da T_1 max

2.14 $V_i = \quad = U_1W_{1i} + U_2W_{2i} + U_3W_{3i}$

$V_1 = 1 = 1*1 + 0*1 + 0*1$

$V_2 = 0 = 1*0 + 0*1 + 0*1$

$V_3 = 1 = 1*1 + 0*1 + 0*1$

Phase 2: Erkennungsphase

Da die Anpassung der Gewichte abgeschlossen ist, kann ein neuer Eingabevektor angelegt werden:

3.1 Lege neuen Eingabevektor $\vec{I} = (011)$ an

3.2 $g_2 = 1$, da $U \neq \vec{0}$

3.3 $S_i = \quad (I_i * V_i) \vee (g_1 * V_i) \vee (I_i * g_1)$

$S_1 = 0 = (0*1) \vee (0*1) \vee (0*0)$

$S_2 = 0 = (1*0) \vee (0*0) \vee (1*0)$

$S_3 = 1 = (1*1) \vee (0*1) \vee (1*0)$

3.4 $R = 1$, da $\frac{\sum_i^m S_i}{\sum_i^m I_i} = \frac{1}{2} = 0,5 < 0,7$

Phase 4: Suchphase

3.5 Da der Reset F_2 blockiert , wird T_j nicht benötigt

3.6 $U = (000)$, da $R = 1$

3.7 $g_1 = 1$, da $I \neq \vec{0} \wedge \vec{U} = \vec{0}$

3.8 $V_i = \quad = 0*W_{1i} + 0*W_{2i} + 0*W_{3i}$

$\vec{V} = (000)$

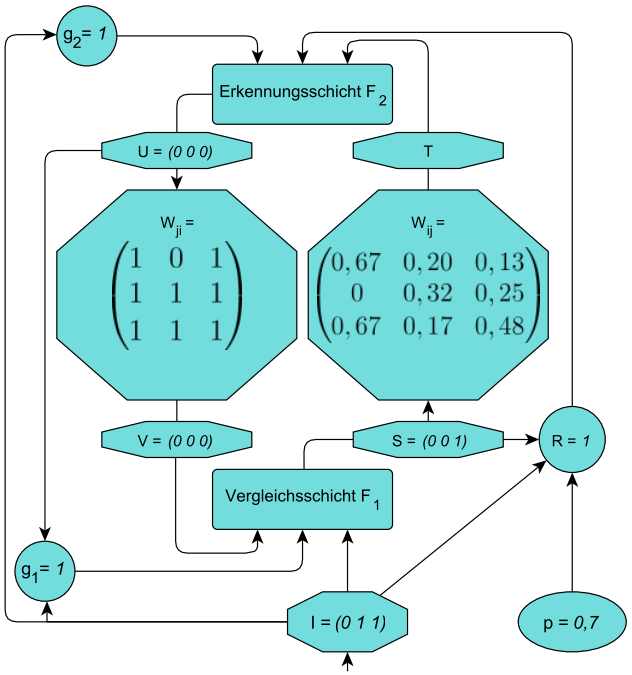

Abbildung 6: *Schritt 3.1-3.7*

Anschließend stellt $\vec{S}$ wieder ein Duplikat von $\vec{I}$ dar. Nun kann T_j berechnet werden um mit der nächsten Kategorie fortzufahren. Diese Schritte laufen analog der beim vorherigen Vektor ab und werden aus diesem Grund nicht mehr einzeln behandelt.

4 Schluss

Die Methodik des ART-1 ist eine Möglichkeit für unüberwachtes Lernen. Sie klassifiziert selbständig binäre Muster in eine zuvor festgelegte Zahl an Mustern, ohne vorherige Festlegung des Aufbaus einzelner Muster. Obwohl der ART-1 ein bereits etwas älteres Verfahren ist und nur binär sortiert, sollte er gewissermaßen als Grundlage für seine moderneren Ansätze, wie die des schnelleren ART-2A, des Fuzzy ART oder des ARTMAP, welcher mit überwachten Lernverfahren arbeitet, angesehen werden. Die Besonderheit hierbei ist die erfolgreiche Lösung des Stabilitäts-Plastizitäts-Dilemmas, welche auf der Tatsache beruht, dass eine vorhandene Kategorie nur bei entsprechender Ähnlichkeit korrigiert bzw. verbessert wird. So wird die bereits vorhandene Kategorie größtenteils beibehalten, ein neues Muster aber gleichzeitig implementiert.

Literatur

[1] N. Hoffmann. *Kleines Handbuch Neuronale Netze*. Vieweg, Wiesbaden, 1st edition, 1993.

[2] R. Moore. Adaptive Resonance Theorie 1 (ART-1). Website, 2002. Available online at `http://wwwmath.uni-muenster.de:8010/Professoren/Lippe/lehre/skripte/opt/ART-Architekturen/ART1.html`; visited on June 6th 2012. Translated by Suleyman Isik on 2002-08-27.

[3] S. Nusser. Adaptive Resonance Theory (ART). Website, 2004. Available online at `http://www.seb-nusser.de/studium/skripte/AdaptiveResonanceTheory.pdf`; visited on June 6th 2012.

[4] G. Ray. *Neuronale Netze eine Einführung in die Grundlagen, Anwendungen und Datenauswertung*. Huber, Bern, 1st edition, 2008.

[5] A. Zell. *Simulation neuronaler Netze*. Oldenbourg, München, 3rd edition, 2000.

A Vorgehen ART-1

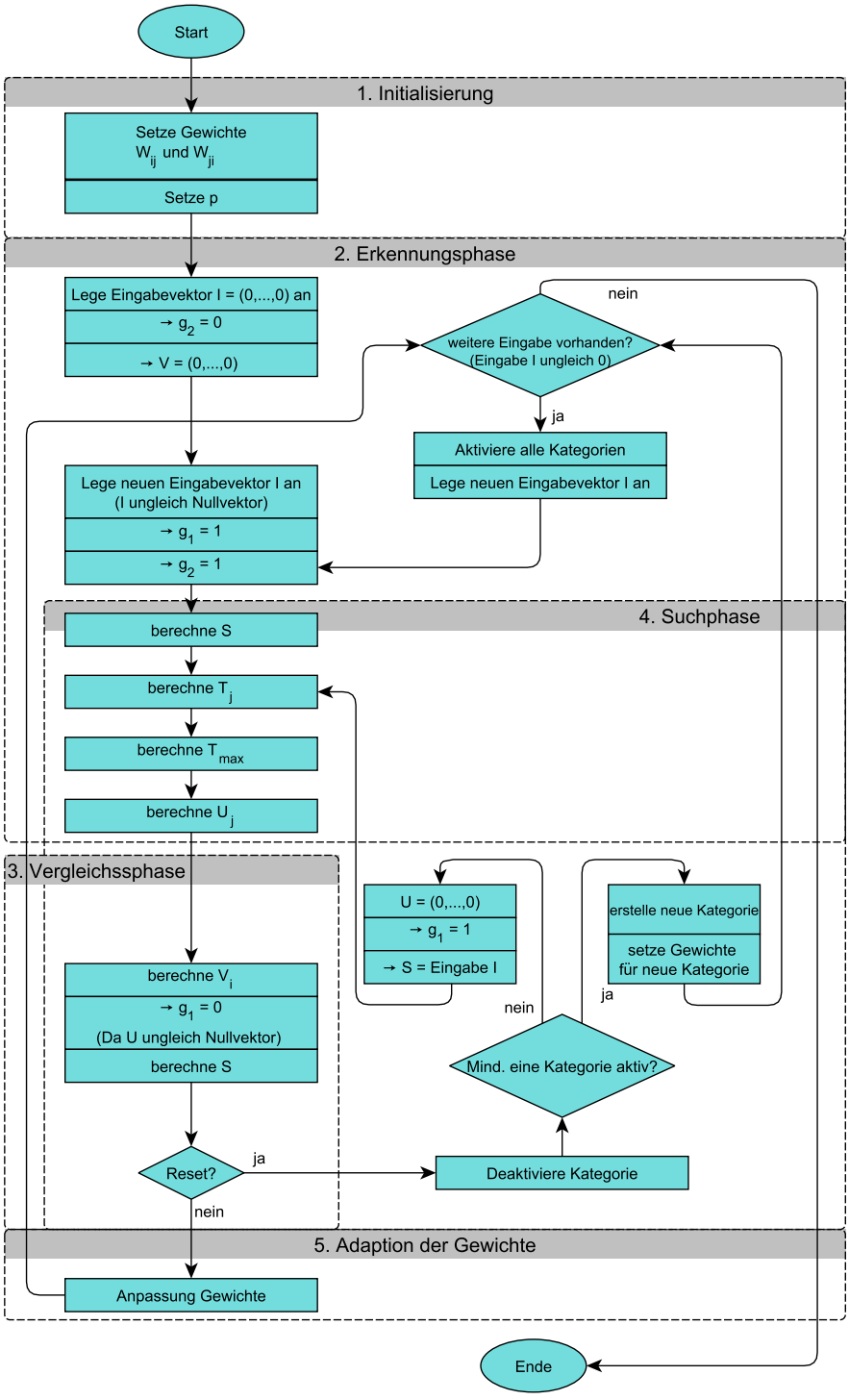

Abbildung 7: Vorgehen des ART-1