Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

HAUPTSEMINAR

Anticipatory Classifier Systems

vorgelegt von	:	Christian Bornträger
geboren am	:	09.05.1978
E-Mail Adresse	:	christian@borntraeger.net
Telefon	:	0177 / 5152290
Studiengang	:	Ingenieurinformatik/M97
Studienrichtung	:	Multimediale Informations-
	:	und Kommunikationssysteme
Matrikel-Nr.		26092

Betreuender Hochschullehrer	:	Dipl.-Inf. Andrea Heinze
Abgabetermin	:	30.01.2002

Ilmenau, 31. Januar 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Wolfgang Stolzmann führte mit den Anticipatory Classifier Systems eine weitere Form der Learning Classifier Systems ein.
Diese unterscheiden sich insbesondere dadurch, dass sie nicht nur Zustands-Handlungs-Verbindungen lernen sondern auch das Ergebnis vorwegnehmen.
Deshalb enthalten die Klassifikatoren neben den Bedingungen und der Aktion außerdem noch einen Erwartungsteil.
Das eigentliche Lernen geschieht dabei durch Anpassen der Klassifikatoren. Dabei gibt es 3 Möglichkeiten:

- die Stärke/Qualität des Klassifikators wird verändert
- es wird ein neuer Klassifikator aus bestehenden abgeleitet
- ein schwacher Klassifikator wird gelöscht

Die Anpassung der Klassifikatoren beruht dabei auf dem Vergleich der Erwartungen mit dem tatsächlichen Zustand der Umwelt.
Stimmt die Erwartung nicht mit dem tatsächlichen Ergebnis überein, erhält dieser eine Marke, welche den Startzustand enthält der zum Fehler führte. Ist es außerdem nicht möglich einen Klassifikator abzuleiten,der korrekt antizipiert, wird die Qualität des Klassifikators gesenkt (*not correctable case*).
Es wird versucht, neue Klassifikatoren aus bestehenden zu bilden.
Dabei werden 2 verschiedene Methoden benutzt. Einerseits werden Klassifikatoren aus fehlerhaften Klassifikatoren gebildet (*case of correctable assumptions and expectations*). Andererseits werden auch neue Klassifikatoren aus alten abgeleitet wenn diese korrekt waren, jedoch zu einem früheren Zeitpunkt fehlerhaft antizipierten (*specifikation of unchanging components*).
Ändert die Aktion den sensorischen Eindruck nicht, wird der Klassifikator, unabhängig ob korrekt oder nicht korrekt antizipiert wurde, abgeschwächt(*useless case*).
War die Erwartung erfolgreich und änderte den Zustand der Umwelt,wird der Klassifikator verstärkt (*expected case*).
Ein großes Problem vieler Classifier Systems sind Alias-Zustände. Stolzmann löst dieses Problem durch Aktionsketten. Grundidee ist das Überspringen dieser Zustände.
Dabei werden der Klassifikator der korrekt in den Alias-Zustand führte sowie ein Klassifikator der korrekt aus den Alias-Zustand heraus führte, verknüpft.
Grundziel der Anticipatory Classifier Systems ist das nachempfinden kognitiver Lernprozesse. Dabei wird davon ausgegangen, dass sich Lebewesen von der Umgebung ein internes Modell bilden, um die Auswirkungen von Handlungen vorherzusagen und somit die Handlungsplanung zu verbessern.
Als internes Modell gilt in Anticipatory Classifier Systems die Menge der sicheren Klassifikatoren. Dies sind alle Klassifikatoren deren Qualität eine Schwelle übersteigt.
Anhand dieser sicheren Klassifikatoren kann ein Anticipatory Classifier System mittels Suche seine Handlungen so ausrichten, dass ein Zielzustand möglichst schnell erreicht wird.

1 Einführung

An der Universität Würzburg versucht das Institut für Psychologie eine Schnittstelle zwischen künstlicher Intelligenz und kognitiver Psychologie zu schaffen [pre].
Basierend auf dem Lernmechanismus der „antizipativen Verhaltenssteuerung“von Prof. Dr. Joachim Hoffmann [Hof93] wurde dieses Projekt ins Leben gerufen.
Menschen und Tiere lernen, sich so zu verhalten dass sie bestimmte Ziele erreichen. Dafür lernen sie welches Verhalten zu welchen Effekten führt.
Die Grundidee dieses Lernens ist das Vorwegnehmen von Auswirkungen einer Aktion. Die Vermutung wird danach mit dem tatsächlichen Ergebnis verglichen.
Ziel dieses Projektes ist es, den biologischen Mechanismus für die künstliche Intelligenz nutzbar zu machen. Stolzmann [Sto] gelang es, den Lernmechanismus zu dem Lernalgorithmus „Anticipative Classifier Systems“ umzusetzen.

2 Ursprünge in der Psychologie

Stolzmann [Sto] beschreibt als Grundlage für seine Anticipative Classifier Systems (ACS) eine Lerntheorie, welche vom Psychologen Tolman [Tol32] beschrieben und von Hoffmann formell umgesetzt wurde.
Dabei ist eine Belohnung des Systems nicht nötig, das System lernt latent aus den Auswirkungen.
Sei ρ_t die Belohnung zum Zeitpunkt t, dann kann ein ACS auch für

$$\rho_t = 0, \forall t \tag{1}$$

ein Modell der Umgebung entwickeln.
Das Grundprinzip der Antizipativen Verhaltensteuerung ist in (2) dargestellt.

$$\underbrace{S_{start} \longleftrightarrow R \overbrace{\longrightarrow E_{ant} > Verg}^{reinforcement} leich < E_{real}}_{Spezialisierung} \tag{2}$$

Dabei werden 4 Grundannahmen Hoffmanns deutlich:

- jede Aktion R ist mit einer Erwartung der Auswirkungen E_{ant} verknüpft
- die Verbindungen zwischen Aktionen und erwarteten Wirkungen werden durch Reinforcement gewonnen bzw. verstärkt
- wenn die Aktion-Erwartungs-Beziehung nur in bestimmten Startzuständen S_{start} zutrifft, werden die Zustandsmerkmale dahingehend spezialisiert
- Reinforcement der Erwartung und Spezialisierung der Startzustände bedürfen eines Vergleiches zwischen erwarteten und tatsächlichen Wirkungen

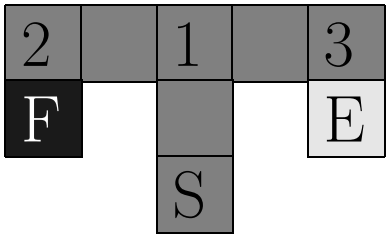

Abbildung 1: Rattenexperimente im T-Labyrinth zeigen latentes Lernverhalten

Stolzmann bezieht sich außerdem auf Experimente die Edward Seward [Sew49] 1949 durchführte. Dabei wurden Ratten in ein einfaches Labyrinth (Abbildung 1) entlassen. Nach 24 Stunden ohne Nahrung wurden sie in eine der Zielboxen (F oder E) gesteckt. In einer befand sich Futter, in der anderen nicht.
Eine Minute später wurden die Ratten zurück in die Startbox (S) versetzt. 28 von 32 Ratten fanden sofort den Weg in die Box mit dem Futter.
Dieses Experiment sollte beweisen, dass Ratten in der Lage sind latent zu lernen, also ein Modell der Umgebung zu entwickeln, ohne eine Belohnung zu erhalten.
Stolzmann zeigt Simulationen in denen ACS vergleichbares leisten [SBHG00].

3 Umsetzung des antizipativen Lernens

3.1 Grundlagen

Anticipatory Classifier Systems basieren auf den von John Holland vorgestellten Classifier Systems bzw. deren Erweiterung Learning Classifier Systems [KL99] [lcs].
Das Grundprinzip der Classifier Systems ist ein Regelwerk, die sogenannten Classifier bzw. Klassifikatoren.
Classifier Systems sind neben dem Q-Learning häufig eingesetzte Methoden zum Reinforcement-Lernen.
Dabei sind die Klassifikatoren Regeln, die je nach Zustand zutreffen oder nicht zutreffen und bei Aktivierung bestimmte Aktionen zur Folge haben.
Sie teilen bzw. klassifizieren den Eingangsraum in verschiedene Zustände.
Die Klassifikatoren lernen dabei anhand der erhaltenen Belohnung.

3.2 Erweiterung

Stolzmann erweitert die Klassifikatoren in seinen Anticipatory Classifier Systems (ACS) dabei um einen Wirkungsteil. Der Wirkungsteil des Klassifikators antizipiert die Auswirkung der getroffenen Handlung auf die Umwelt, er nimmt sie also vorweg.
Dies ist der Hauptunterschied zu anderen Classifier Systems. In diesen wird die Auswirkung auf die Umwelt nicht betrachtet, einzig die Belohnung oder Bestrafung der Aktion führt zu Lerneffekten.
Ein ACS besteht aus 4 Komponenten:

- einem Eingabe-Interface, mit Sensoren zum Erfassen der Zustände der Umwelt σ, die in Messwerte S_t umgesetzt werden
- einem Ausgabe-Interface mit Aktoren zum Ausführen der Aktionen α.
- einer Klassifikatoren-Tabelle, die die Regeln enthält
- einer Message-Liste zur Zwischenspeicherung

3.3 Repräsentation der Daten in der Klassifikatoren-Tabelle

In ACS wird das Wissen in einer Tabelle von Klassifikatoren repräsentiert. Diese Klassifikatoren c bestehen aus 3 Teilen:

- die Bedingungen **C** *(conditions)*: Sie geben den Zustand der Umgebung bzw. die Werte der Sensoren wieder, für die der Klassifikator zutreffen soll. Dabei gilt

$$C \in \{0, 1, \#\}^n \tag{3}$$

  - 0 steht für Sensor spricht nicht an
  - 1 steht für Sensor spricht an
  - # bedeutet, dass die Gültigkeit des Klassifikators nicht von diesem Sensor abhängt
  - n ist die Anzahl der Sensoren

- die Handlung **A** *(action)*: Sie gibt wieder, welche Aktoren, angesprochen werden sollen. Die Aktoren üben die Wirkung α auf die Umwelt aus.
- die Erwartung **E** *(expectation)*: Sie ist der vermutliche Zustand nach einer Handlung. Es gilt:

$$E \in \{0, 1, \#\}^n \tag{4}$$

  - 0 die Eigenschaft der Umgebung wird sich zu 0 ändern
  - 1 die Eigenschaft der Umgebung wird sich zu 1 ändern
  - # die Eigenschaft der Umgebung wird sich nicht ändern

Ein Klassifikator ist die Verknüpfung der Teile C-A-E. Als Startpunkte für die Klassifikatorentabelle werden allgemeine Regeln angenommen, die jede mögliche Aktion repräsentieren (Tabelle 1).
Das Prinzip kann problemlos auf eine diskrete, endliche Menge erweitert werden:

$$C, E \in \{Zeichenvorrat, \#\} \tag{5}$$

Jeder Klassifikator hat außerdem noch zwei Maßzahlen:

- die Qualität q *(quality)*, welche die Genauigkeit der Vorhersage schätzt

C	A	E
########	$\Leftarrow$	########
########	$\Rightarrow$	########
########	$\Downarrow$	########
########	$\Uparrow$	########

Tabelle 1: Start-Klassifikatoren für 8-dimensionale Eingangsvektoren und 4 mögliche Aktionen

- die Belohnung r *(reward-prediction)*, welche die Belohnung der Umgebung vorhersagt

Für das latente Lernen der Umwelt mittels antizipativem Verhalten ist r nicht notwendig. Im Gegenteil, Stolzmann zeigt mit einigen Simulationen [Sto], dass durch eine große Belohnung für das Erreichen des Zielzustandes die Belohnung wesentlich stärker in die Auswahl eingeht als die Qualität (Exploration - Exploitation -Problem).
Dabei strebt das System möglichst schnell in Richtung Zielzustand, erforscht aber die Umgebung nur in geringerem Maße.
Die Handlungsplanung kann also anhand einer starken Belohnung r im Zielzustand oder durch weitere Verfahren erreicht werden (Abschnitt 5).

3.4 Antizipieren

Antizipieren bedeutet in diesem Zusammenhang vorhersagen.
Das Grundprinzip eines ACS besteht darin, je nach Zustand Handlungen auszuführen, und den neuen Zustand vorherzusagen (zu antizipieren). Je nachdem ob die Vorhersage korrekt war, werden die internen Regeln angepasst, so lernt das System.
Die Funktionsweise beruht auf einer Message-Liste. Sie dient zur Kommunikation zwischen den Komponenten und zur Zwischenspeicherung.
Zur Zeit t erzeugen die Sensoren eine Wahrnehmung der Umwelt S_t (Abbildung 2).
Die aktuelle Wahrnehmung der Umwelt, repräsentiert durch den Vektor S_t wird auf der Message-Liste gespeichert und mit den vorhandenen Klassifikatoren elementweise verglichen.
Damit ein Klassifikator matcht, muss jedes Element c der Klassifikator- Bedingung C identisch zum passenden Element s des Zustandsvektors S_t sein.
Alternativ darf das Element c auch das #-Symbol sein.
Alle Klassifikatoren auf die dies zutrifft bilden das „Match-Set“.
Danach wird ein passender *(matching)* Klassifikator aus dem Match-Set ausgewählt (Abbildung 3).
Sind mehrere Klassifikatoren im Match-Set enthalten, geschieht die Auswahl nach einem gewichteten Zufallsprinzip (Gleichungen 6 und 7).
Die Fitness eines Klassifikators ist definiert als

$$f_c(t) = q_c(t) \cdot r_c(t) \tag{6}$$

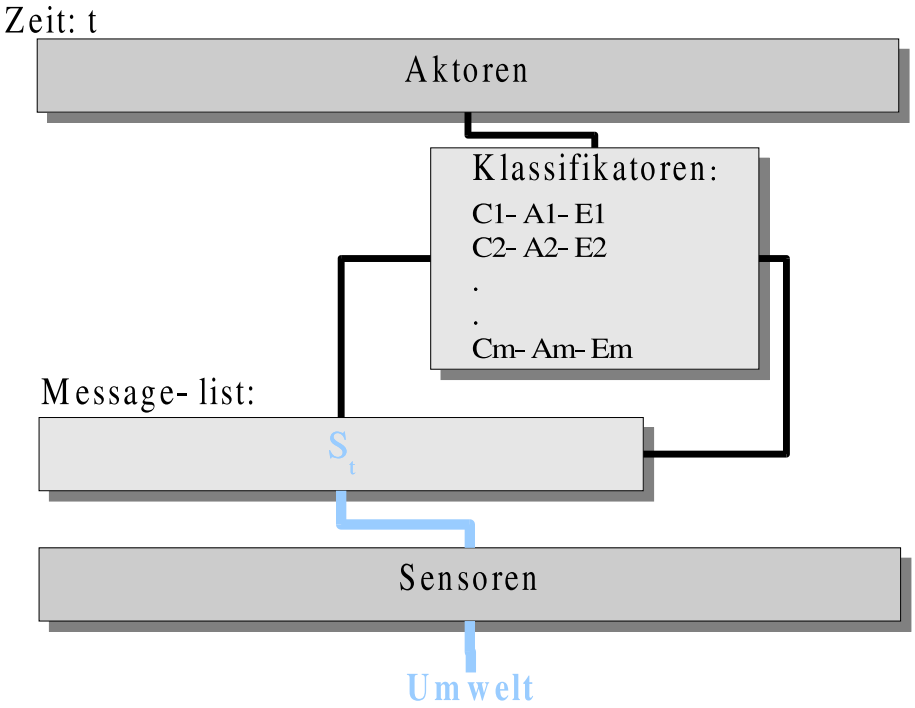

Abbildung 2: Ein sensorischer Eindruck wird auf der Message-Liste gespeichert

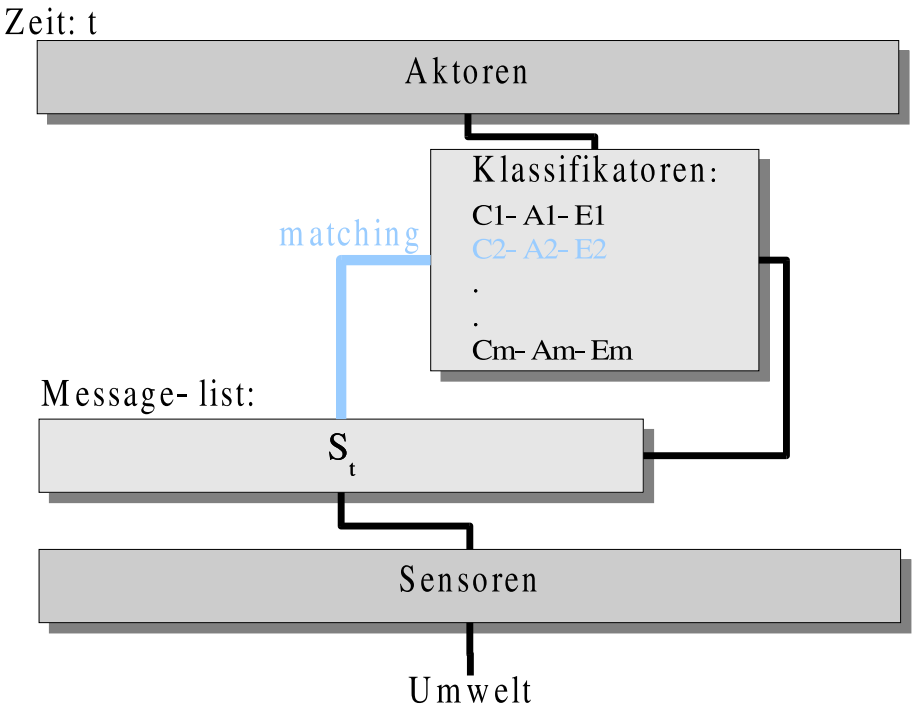

Abbildung 3: Ein passender Klassifikator wird ausgewählt

Es gehen dabei die Qualität q und die erwartete Belohnung r in die Auswahl ein. Die Fitness spielt eine Rolle für die Wahrscheinlichkeit dass ein Klassifikator ausgewählt wird:

$$p(Klassifikator = i) = \frac{f_{c_i}}{\sum\limits_{c_{matching}} f_c} \tag{7}$$

Die Auswahl geschieht dabei mittels Wahl einer Zufallszahl $\in (0, 1)$ welche den Klassifikator bestimmt. Die Klassifikatoren nehmen je nach Wahrscheinlichkeit einen entsprechend großen Bereich zwischen 0 und 1 ein.
Je fitter ein Klassifikator also ist, desto größer ist die Wahrscheinlichkeit, dass der Klassifikator gewählt wird.
Sollte ohne Belohnung gelernt werden, gilt

$$f_c(t) = q_c(t) \tag{8}$$

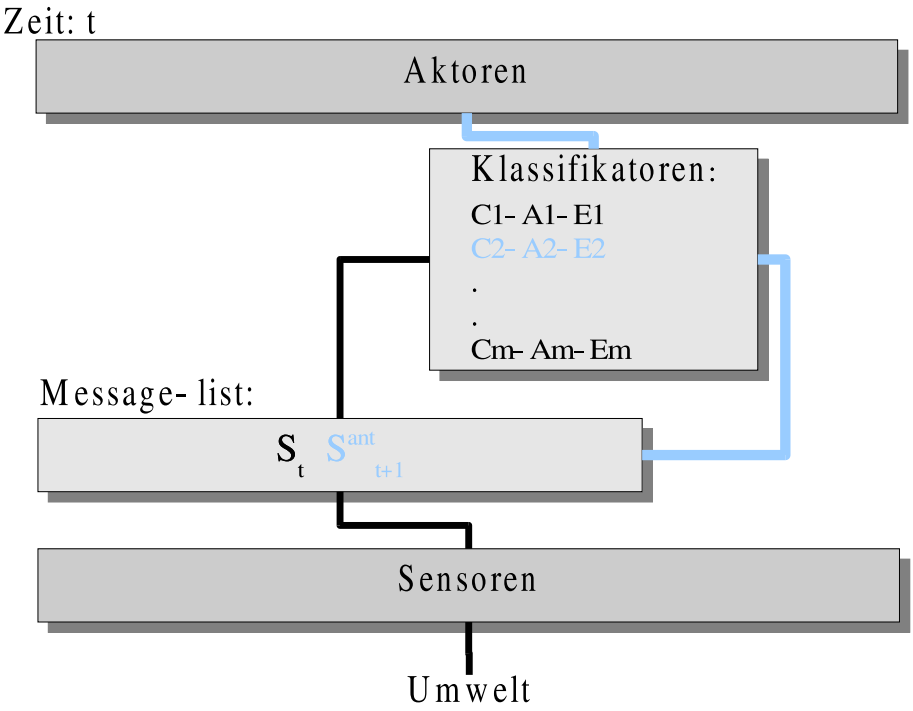

Abbildung 4: Der gewählte Klassifikator wird aktiv

Der gewählte Klassifikator wird aktiv (Abbildung 4) und veranlasst, dass die Aktoren die Aktion A ausführen. Außerdem schickt er eine Nachricht an die Message-Liste, welche den erwarteten neuen Zustand

$$S_{t+1}^{ant} = passthrough(S_t, Ej) \tag{9}$$

beschreibt.

passthrough-Funktion

$passthrough(Argument_1, Argument_2)$ ist dabei eine Funktion, welche die #-Elemente von $Argument_2$ (hier E) durch die korrespondierenden Elemente von $Argument_1$ (hier S_t) ersetzt. Die restlichen Elemente entsprechen denen von $Argument_2$.

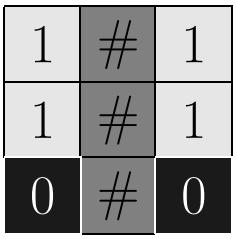

Abbildung 5: Klassifikator-Bedingung C

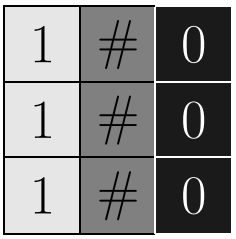

Abbildung 6: Klassifikator-Erwartung E

Seien die Abbildungen 5 und 6 C und E des Klassifikators c:

$$c = \underbrace{\begin{matrix}1 & \# & 1\\ 1 & \# & 1\\ 0 & \# & 0\end{matrix}}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\begin{matrix}1 & \# & 0\\ 1 & \# & 0\\ 1 & \# & 0\end{matrix}}_{Erwartung\ E} \tag{10}$$

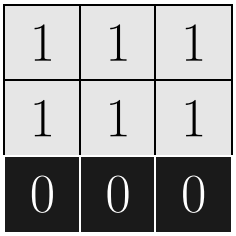

Abbildung 7: Beispielzustand S_t

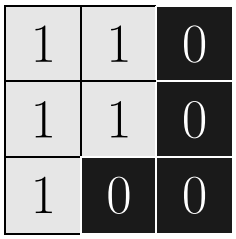

Abbildung 8: Antizipierter $ZustandS_{t+1}^{ant}$

Aus dem Zustand (Abbildung 7)

$$\begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 0 & 0 & 0\end{matrix}$$

würde nach Gleichung (10) Zustand

$$\begin{matrix}1 & 1 & 0\\ 1 & 1 & 0\\ 1 & 0 & 0\end{matrix}$$

(Abbildung 8) antizipiert.

3.5 Lernen

Ein ACS soll sich vor allem ein Modell von der Umwelt bilden. Dazu muss es sein Wissen an die tatsächliche Umwelt anpassen.
Ein neuer Zustand der Umwelt S_{t+1} werde von den Sensoren erfasst (Abbildung 9).

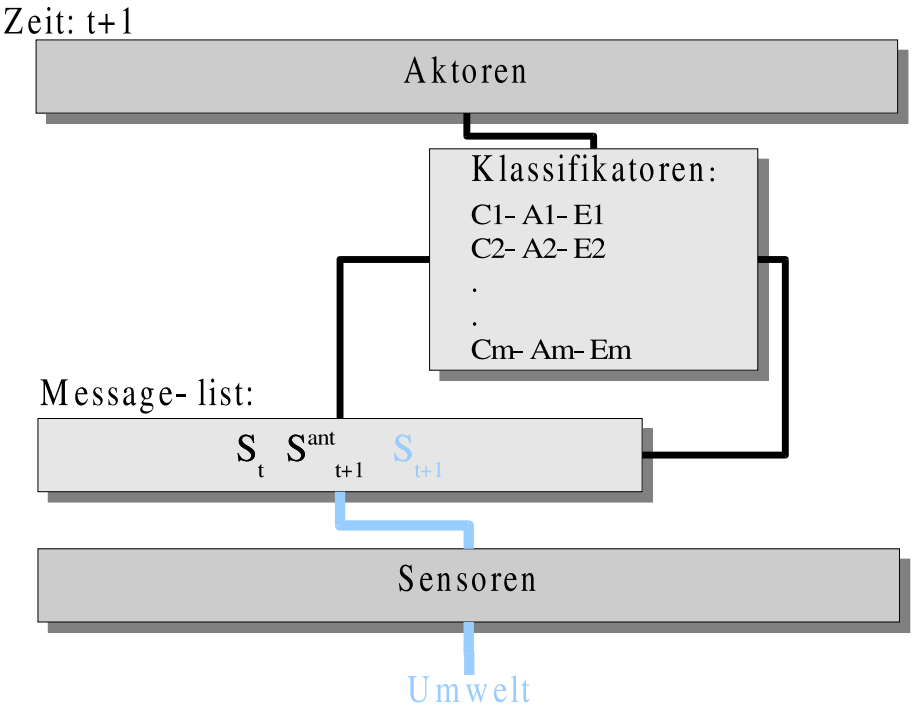

Abbildung 9: Ein neuer sensorischer Eindruck wird auf der Message-Liste gespeichert

Erst durch den Vergleich des antizipierten Zustandes S_{t+1}^{ant} mit dem tatsächlichen Zustand S_{t+1}^{real}(Abbildung 10) können die Regeln angepasst werden - das System lernt.

Dabei kann der Vergleich verschiedene Möglichkeiten ergeben.

3.5.1 Korrekte Antizipation

Wenn die ausgeführte Aktion den sensorischen Eindruck der Umwelt änderte und die Antizipation korrekt war(*expected case*), wird die Qualität des gewählten Klassifikators verstärkt, z.B. nach

$$q_{t+1} = (1 - b_q) \cdot q_t + b_q \tag{11}$$

Dabei sei b_q eine Lernkonstante, mit $b_q \in [0, 1]$.
Hatte die gewählte Aktion jedoch keinen Einfluss auf den sensorischen Eindruck, d.h. $S_t = S_{t+1}$ dann wird die gewählte Aktion A als nicht sinnvoll erachtet (*useless case*). Deshalb wird die Qualität des gewählten Klassifikators gesenkt:

$$q_{t+1} = (1 - b) \cdot q_t \tag{12}$$

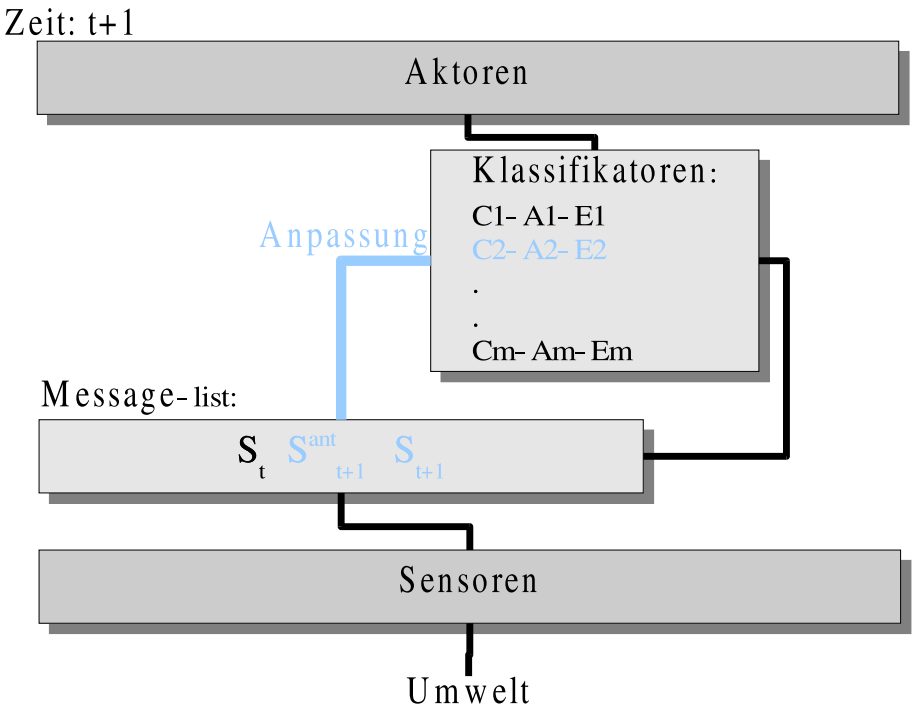

Abbildung 10: Aus neuem Zustand und antizipiertem Zustand werden Lernschritte abgeleitet

Dies kann dazu führen, dass Klassifikatoren in langen konstanten Gängen stark an Qualität verlieren.

3.5.2 Falsche Vorhersage

In dem Fall einer falschen Vorhersage besteht noch die Möglichkeit, aus bestehenden Klassifikatoren neue abzuleiten, die die Veränderung der Umwelt korrekt antizipieren. Dafür ist es notwendig, dass die nicht korrekten Elemente des gewählten Klassifikators sowohl für C als auch E #-Symbole sind(*case of correctable assumptions and expectations*).
Nur so können diese ersetzt werden, um einen korrekten Klassifikator zu bilden.
In diesem neuen Klassifikator, der aus dem vorherigen abgeleitet wird, werden die fehlerhaften C-Elemente durch die entsprechenden Elemente aus S_t ersetzt, während die E-Elemente durch die korrespondierenden Elemente aus S_{t+1} ersetzt werden.

- sei $S_t = \mathbf{1}000$
- der gewählte Klassifikator sei

$$c_{alt} = \underbrace{\#\#00}_{Bedingung\ C} \overset{Aktion\ A}{\Uparrow} \underbrace{\#\#11}_{Erwartung\ E} \tag{13}$$

- das System antizipiert also $S_{t+1}^{ant} = 10\mathbf{11}$
- S_{t+1}^{real} sei aber 0011

Der Klassifikator hat also nicht korrekt antizipiert:

$$\mathbf{1}011 \leftrightarrow \mathbf{0}011 \tag{14}$$

Da der Klassifikator korrigierbar ist, wird ein neuer Klassifikator

$$c_{neu} = \underbrace{\mathbf{1}\#00}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\mathbf{0}\#11}_{Erwartung\ E} \tag{15}$$

erzeugt, der den Zustandswechsel korrekt antizipiert. Dabei wird der alte Klassifikator c_{alt} nicht gelöscht.
Für den Fall, dass die nicht korrekten Elemente des gewählten Klassifikators keine #-Symbole sind (*not correctable case*) kann aus dem bestehenden Klassifikator kein neuer Klassifikator gebildet werden.
Deshalb wird die Qualität des alten Klassifikators gesenkt:

$$q_{t+1} = (1 - b) \cdot q_t \tag{16}$$

3.6 Vergessen

Bis jetzt werden nur neue Klassifikatoren erzeugt. Deshalb ist ein Mechanismus notwendig, welcher unzuverlässige Klassifikatoren löscht. Dafür wird ein Schwellwert

$$\delta^{delete} \in [0, 1] \tag{17}$$

definiert, typischerweise 0.10.
Fällt die Qualität eines Klassifikators unter diesen Schwellwert, gilt er als unzuverlässig und wird gelöscht.
Trifft ein Klassifikator zu oft auf unpassende Zustände, kann es passieren, dass er fälschlicherweise gelöscht wird, obwohl aus ihm in anderen Teilen Umwelt evtl. sehr gute Klassifikatoren abgeleitet werden könnten.
Einzig die Start-Klassifikatoren mit $C \in \#$ und $E \in \#$ werden nie gelöscht, da sie notwendig sind, um komplett neue Situationen zu lernen.

3.7 Spezifizierung von konstanten Elementen

Bei den bisherigen Algorithmen werden neue Klassifikatoren nur im Fall einer falschen aber korrigierbaren Vorhersage erzeugt (*case of correctable assumptions and expectations*).

Dies bedeutet, dass sich für die Spezialisierung ein #-Element des Klassifikators von S_t nach S_{t+1} ändern muss. Sei der gewählte Klassifikator

$$c_{alt=} \quad \underbrace{\#\#10}_{Bedingung\ C} \quad \overbrace{\Uparrow}^{Aktion\ A} \quad \underbrace{\#\#11}_{Erwartung\ E} \tag{18}$$

Die vermuteten und tatsächlichen Auswirkungen der Handlung ⇑ auf eine Beispielumwelt werden in Tabelle 2 dargestellt.

Startzustand	antizipierter Zustand	Endzustand
0010	00**11**	00**00**
0110	0**111**	0**000**
1010	1011	1011
1110	1111	1111

Tabelle 2: Zustand und Folgezustand einer Beispielumwelt für die Aktion ⇑

Die ersten beiden Fälle wären nicht korrigierbare Fälle, die Qualität des Klassifikators müsste nach Gleichung (16) gesenkt werden aber der dritte und der vierte Fall wären ein erwarteter Fall.
Die Qualität würde nach Gleichung (11) erhöht. Dies führt offensichtlich zu keinem korrekten, stetigen Lernen.
Deshalb führte Stolzmann so genannten Marken ein. Mit diesen merken sich Klassifikatoren einen Zustand in dem sie nicht korrekt antizipieren und dies unkorrigierbar ist.
Wird dieser Klassifikator beispielsweise im Zustand 0010 angewandt, stellt das ACS fest, dass der Klassifikator nicht korrekt antizipierte. Deshalb bekommt er den Startzustand als Marke:

$$c_{alt} = \underbrace{\#\#10}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\#\#11}_{Erwartung\ E} \overbrace{\mapsto (0,0,1,0)}^{Marke\ M} \tag{19}$$

Wird der Klassifikator nun im Zustand 1110 korrekt angewandt vergleicht das ACS den Startzustand mit der Marke.

$$\overbrace{\mathbf{00}10}^{Marke\ M} \leftrightarrow \overbrace{\mathbf{11}10}^{Startzustand\ S} \tag{20}$$

Werden Unterschiede gefunden, wird ein Element ausgewählt, das #-Symbol im C als auch im E-Teil des Klassifikators wird durch Startzustand und Endzustand ersetzt.
Die Position des Elementes in C als auch in E muss die gleiche sein. Als neuer Klassifikator kann also

$$c_{neu_1} = \underbrace{\mathbf{1}\#10}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\mathbf{1}\#11.}_{Erwartung\ E} \tag{21}$$

oder

$$c_{neu_2} = \underbrace{\#\mathbf{1}10}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\#\mathbf{1}11.}_{Erwartung\ E} \tag{22}$$

erzeugt werden. Die Entscheidung welcher der beiden Klassifikatoren erzeugt wird ist rein zufällig.

Generalisierte Marken

Als Erweiterung können Marken auch generalisiert werden. Damit können sie mehrere nicht zutreffende Zustände speichern. Dies gelingt mittels einer Oder-Verknüpfung der Marke mit dem Startzustand einer nicht korrekten Antizipation.
Würde der Klassifikator in den Zuständen 0010 und 0110 nicht korrekt vorhersagen, sähe die Marke folgendermaßen aus:

$$(0,0,1,0) \cup (0,1,1,0) = (0,(0,1),1,0) \tag{23}$$

Der oben genannte Algorithmus zur Spezifizierung von Konstanten läuft danach genauso ab. Dabei sei eine Element s des Zustandes S verschieden zu einem Element m der Marke M wenn $s \notin m$.
Trifft der Klassifikator

$$c_{alt} = \underbrace{\#\#10}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\#\#11}_{Erwartung\ E} \overbrace{\mapsto (\mathbf{0},(0,1),1,0)}^{Marke\ M} \tag{24}$$

auf den Zustand **1**010 so kann nur ein Folgeklassifikator erzeugt werden:

$$c_{neu} = \underbrace{\mathbf{1}\#10}_{Bedingung\ C} \overbrace{\Uparrow}^{Aktion\ A} \underbrace{\mathbf{1}\#11}_{Erwartung\ E} \tag{25}$$

Meiner Meinung nach ist hieraus ersichtlich, dass das Lernen repräsentationsabhängig ist.
Schwerwiegender ist jedoch die Fehlerhaftigkeit dieses Verfahrens. Seien die Zustände in denen ein Klassifikator falsch antizipiert hätte 0101 **und** 1001**. Die Marke wäre also**

$$((0,1),(0,1),0,1) \tag{26}$$

Somit würde ein ACS die Zustände 0001 **und** 1101 **als Teil der Marke interpretieren, was jedoch nicht der Fall ist.**

0	0	0	0	0	0	0	1
1	0	1	S_3	1	0	1	1
1	S_1	S_2	0	S_4	S_5	S_6	0
0	0	0	0	0	0	0	1

Abbildung 11: Labyrinth mit alias-Zuständen

3.8 Alias-Zustände, nicht-Markov-Umgebungen

Alias-Zustände sind Zustände, die unterschiedlich sind, jedoch den gleichen (beschränkten) sensorischen Eindruck vermitteln.
Die sensorischen Eindrücke zu S_1 und S_5 in Abbildung 11 sind identisch

$$S_{1,5} = \begin{matrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{matrix} \tag{27}$$

.
Trotzdem führt eine Rechtsbewegung aus S_1 in einen anderen Zustand als aus S_5.
Ohne Nutzung weiterer Informationen ist es für das System unmöglich, beide Zustände vor Ausführung der Aktion zu unterscheiden.
Um diese Zustände nach der Ausführung der Aktion zu erkennen, bedient sich Stolzmann der Spezifizierung von konstanten Elementen. Schlägt diese fehl, d.h. gibt es keine Unterschiede zwischen aktuellem Startzustand und gespeicherter Marke obwohl der Klassifikator einmal falsch und einmal richtig antizipierte handelt es sich um einen Alias-Zustand.
Man kann dieses Problem unter anderem durch Hinzufügen von Speicher lösen, wie Cliff & Ross [CR94] bei den ZCS.

3.9 Aktionsketten

Stolzmann entschied sich für eine Variante in der Aktionen gebündelt werden (*action chunking*).
Dabei werden mehrere Aktionen so verknüpft, dass Alias-Zustände nach Möglichkeit übergangen werden.
Der Grundgedanke ist die Verknüpfung der Klassifikatoren, die in den Alias-Zustand hinein und heraus führen.
Es gebe neben weiteren Klassifikatoren die Klassifikatoren c_1 und c_2.

c_1 sei in S_4 anwendbar:

$$c_1 = \underbrace{\begin{matrix}\# & 1 & 0\\ 0 & 1 & 1\\ \# & \# & \#\end{matrix}}_{Bedingung\ C} \overset{Aktion\ A}{\Rightarrow} \underbrace{\begin{matrix}\# & 0 & 1\\ 1 & 1 & 1\\ \# & \# & \#\end{matrix}}_{Erwartung\ E} \tag{28}$$

c_2 sei anwendbar in S_1 und S_5:

$$c_2 = \underbrace{\begin{matrix}1 & 0 & 1\\ \# & 1 & 1\\ \# & \# & 0\end{matrix}}_{Bedingung\ C} \overset{Aktion\ A}{\Rightarrow} \underbrace{\begin{matrix}0 & 1 & 1\\ \# & 1 & 0\\ \# & \# & 1\end{matrix}}_{Erwartung\ E} \tag{29}$$

Das System befinde sich im Zustand S_1 und der Klassifikator c_2 werde gewählt:

$$S_t = S_1 = \begin{matrix}1 & 0 & 1\\ 1 & 1 & 1\\ 0 & 0 & 0\end{matrix} \tag{30}$$

$$\overbrace{\text{nach Gleichung (29)}}\qquad S^{ant}_{t+1} = S_6 = \begin{matrix}0 & 1 & 1\\ 1 & 1 & 0\\ 0 & 0 & 1\end{matrix} \neq S^{real}_{t+1} = S_2 = \begin{matrix}0 & 1 & 1\\ 1 & 1 & 0\\ 0 & 0 & 0\end{matrix} \tag{31}$$

Nach Gleichung (31) antizipiert c_2 nicht korrekt. Deshalb wird die Qualität nach Gleichung (16) gesenkt und c_2 erhält eine Marke:

$$c_2 = \underbrace{\begin{matrix}1 & 0 & 1\\ \# & 1 & 1\\ \# & \# & 0\end{matrix}}_{Bedingung\ C} \overset{Aktion\ A}{\Rightarrow} \underbrace{\begin{matrix}0 & 1 & 1\\ \# & 1 & 0\\ \# & \# & 1\end{matrix}}_{Erwartung\ E} \overbrace{\mapsto \left(\begin{matrix}1 & 0 & 1\\ 1 & 1 & 1\\ 0 & 0 & 0\end{matrix} = S_1\right)}^{Marke\ M} \tag{32}$$

Nach einiger Zeit sei das System durch andere Regeln im Zustand S_4 und der Klassifikator c_1 werde gewählt.

$$S_t = S_4 = \begin{matrix}1 & 1 & 0\\ 0 & 1 & 1\\ 0 & 0 & 0\end{matrix} \tag{33}$$

$$S^{ant}_{t+1} = S^{real}_{t+1} = S_5 = \begin{matrix}1 & 0 & 1\\ 1 & 1 & 1\\ 0 & 0 & 0\end{matrix} \tag{34}$$

Nach Gleichung (34) antizipiert c_1 korrekt den Zustand S_5.

Jetzt werde Klassifikator c_2 gewählt. Auch dieser antizipiert korrekt den Folgezustand und führt dabei aus dem Zustand S_5 in den Zustand S_6:

$$S_{t+2} = S_6 = \begin{matrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Da c_2 eine gesetzte Marke hat, wird diese mit dem Startzustand S_5 verglichen. Marke (S_1) und S_5 sind identisch. Es handelt sich hierbei um Alias-Zustände.
Da c_1 **korrekt** in den Alias-Zustand $S_{t+1} = S_5$ hinein und c_2 **korrekt** aus den Alias-Zustand heraus führte, können die Klassifikatoren c_1 und c_2 verknüpft werden.
Es ist dabei egal, ob der Klassifikator der in den Alias-Zustand führt, hier also c_1 bereits existierte *(expected case)* oder erst beim Übergang von S_t nach S_{t+1} konstruiert wurde *(specification of unchanging components* oder *case of correctable assumptions and expectations).*
Für das Verknüpfen der Klassifikatoren c_{vorher} und $c_{nachher}$ gibt es dabei folgende Regeln:

$$C_{neu} = passthrough(C_{t+1}, C_t) \tag{35}$$
$$A_{neu} = A_t A_{t+1}(Konkenation) \tag{36}$$
$$E_{neu} = passthrough(E_t, E_{t+1}) \tag{37}$$

Alternativ dazu könnte auch

$$C_{neu} = C_{vorher} \tag{38}$$

genutzt werden. Stolzmann nutzt aber Gleichung (35), da diese unter Umständen speziellere Startzustände bildet, die genauer in die Situation passen. So dass die Aktionskette nur dann ausgeführt wird wenn wirklich ein Alias-Zustand bevor steht.
In diesem Beispiel ist $c_t = c_1$ und $c_{t+1} = c_2$. Es wird also der neue Klassifikator

$$c_{neu} = c_3 = \underbrace{\begin{matrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ \# & \# & 0 \end{matrix}}_{Bedingung\ C} \overset{Aktion\ A}{\Rightarrow\Rightarrow} \underbrace{\begin{matrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ \# & \# & 1 \end{matrix}}_{Erwartung\ E} \tag{39}$$

gebildet.
c_3 in S_4 angewandt überspringt den Alias-Zustand S_5.
Wird c_1 in Zukunft noch im Zustand S_1 angewandt, schlägt die Spezifizierung von konstanten Elementen fehl und die Qualität wird gesenkt.
Langfristig kann es passieren, dass c_1 als unzuverlässig gilt und gelöscht wird.
Es ist möglich, dass eine Aktionskette $A = a_1, \cdots, a_k$ mehr als 2 Aktionen beinhaltet. Deshalb wird eine Maximalanzahl atomarer Aktionen festgelegt:

$$k \leq \alpha_{max} \tag{40}$$

3.10 Kontrollierte Aktionsketten

Durch den vorherigen Algorithmus können nutzlose Aktionsketten entstehen, da dieser **nicht** kontrolliert, ob Aktionen gegenläufig sind. Dies wäre nur möglich, wenn das System noch logisches Verständnis besäße.
Es ist aber nicht in der Lage, die Bedeutung von Aktionen zu erfassen. Deshalb müssen andere Lösungen gefunden werden.
Beispielsweise sei $A \Leftarrow\Rightarrow\Uparrow$.
Ein Algorithmus um dieses Problem zu umgehen, ist folgender:
Die Ausführung einer Aktionskette wird mit einer Liste M kontrolliert:

$$M := \{m_0\} \text{ mit } m_0 := S_t \tag{41}$$

Jede Aktion $a_{i=1\cdots k}$ der Aktionskette wird ausgeführt und das Resultat S_t wird mit M verglichen. Es wird also kontrolliert, ob das ACS im Laufe der Aktionskette bereits im aktuellen Zustand war.
Ist dies der Fall, also $S_t \in M$ dann wird die Qualität q gesenkt und die Ausführung wird gestoppt.
Ist der Zustand hingegen neu, also $S_t \notin M$, dann wird $m_i = S_t$ zu M hinzugefügt:

$$M := M \cup m_i \tag{42}$$

Wurde die Aktionskette A vollständig abgearbeitet, wird $m_k = S_{t+1}$ mit S_{t+1}^{ant} verglichen und der Lernalgorithmus wird wie bisher ausgeführt.
Dieser Algorithmus ist notwendig, um Endlosschleifen zu verhindern.
Meiner Meinung nach verhindert er leider auch die Ausnutzung von Aktionsketten in langen Gängen, wie in Abbildung (12).

0	0	0	0	0	0
1	S_1	S_2	S_3	S_4	S_5
0	0	0	0	0	0

Abbildung 12: Labyrinth mit langen Gängen

4 Lernen mit Belohnung

Auch wenn dies nicht Forschungsschwerpunkt ist, so bieten doch ACS mit „Lernen mittels Belohnung“auch einen zweiten unabhängigen Lernalgorithmus.
Wenn ein ACS von der Umwelt Belohnung $\rho(t)$ erhält, kann man es als normales Classifier System mit Reinforcement-Lernen betrachten.

Dabei wird beispielsweise der „bucket brigade“-Algorithmus [Hol80] verwendet, um die Belohnung r der Klassifikatoren anzupassen. Dieses Verfahren ist ein Verfahren ähnlich dem Q-Learning.
Es lässt sich außerdem zeigen, dass Classifier Systems dem Q-Learning entsprechen sofern man folgende Einschränkungen trifft [DB94]:

- es gibt keine internen Zustände
- es gibt keine don't care-Symbole (#)
- es sind keine Strukturellen Änderungen möglich, d.h. alle möglichen Zustände werden erfasst

Sei c_t der aktive Klassifikator zum Zeitpunkt t und c_{t+1} der aktive Klassifikator zum Zeitpunkt t+1.
Erhält das System eine Belohnung ($\rho_{t+1} \neq 0$) so wird ein Teil der Belohnung ρ an den vorherigen Klassifikator abgeben:

$$r_{c_t}(t+1) = (1 - b_r) \cdot r_{c_t}(t) + b_r \cdot \rho(t+1) \tag{43}$$

Erhält das System dagegen keine Belohnung ($\rho_{t+1} = 0$) wird nach diesem Algorithmus eine Teilmenge von r an den vorherigen Klassifikator übertragen:

$$r_{c_t}(t+1) = (1 - b_r) \cdot r_{c_t}(t) + b_r \cdot r_{c_{t+1}}(t) \tag{44}$$

5 Handlungsplanung

5.1 Modell der Umgebung

Ein ACS lernt in der Erforschungsphase eine interne Repräsentation seiner Umgebung. Nach dieser Erforschungsphase sollte es in der Lage sein, zielgerichtet vorzugehen.
Ist die Qualität eines Klassifikators sehr groß, ist die Wahrscheinlichkeit einer korrekten Antizipation ($S_2^{real} = S_2^{ant}$) ebenfalls sehr groß.
Ein Klassifikator wird als sicher bezeichnet, wenn

$$q_c(t) \geq 1 - \epsilon^{sure} \tag{45}$$

Dabei sei $\epsilon^{sure} \in [0.1]$ eine kleine Konstante.
Ein ACS kann mehrere sichere Klassifikatoren verketten, da es nicht notwendig ist, auf den nächsten Umweltzustand zu warten.
Ein internes Modell der Umgebung ist definiert als Menge aller sicheren Klassifikatoren.

5.2 Finden des Zieles

Das interne Modell kann genutzt werden, um Handlungen zielgerichtet zu planen. Als Ziel gilt dabei das Erreichen eines bestimmten Zustandes.
Dies sollte auf dem schnellsten Wege geschehen.
Alias-Zustände bereiten normalerweise keine Probleme, da Klassifikatoren die in Alias-Zuständen gelten nicht zu den sicheren zählen.
Sollte es beim Lernen aber nicht gelungen sein, diese zu überspringen, schlägt auch die Handlungsplanung fehl.
Der aktuelle Zustand sei S_N und das Ziel sei F_S.
Eine Möglichkeit der Wegeplanung ist die vorwärts gerichtete Breitensuche.

1. Die Liste aller sicheren Klassifikatoren, deren C_i mit dem Startpunkt S_N matcht sei:
$$c_i = (C_i, A_i, E_i^{ant} \text{ für } i = 1, \cdots, n) \tag{46}$$
2. berechne alle $S_1^{ant}, \cdots, S_n^{ant}$ mittels $S_i^{ant} = passthrough(S_N, E_i)$
3. Ist der Zielpunkt $F_S \in \{S_1^{ant}, \cdots, S_n^{ant}\}$, beende die Suche. Ansonsten führe 1. und 2. für $S_1^{ant}, \cdots, S_n^{ant}$ statt S_N aus, bis F_S gefunden oder eine maximale Anzahl von Suchschritten max^{depth} erreicht ist.

Wurde F_S gefunden, führt das ACS alle Handlungen A_i aus.
Dieser Verhalten kann als zielgerichtet interpretiert werden.
Einer weitere Möglichkeit ist die rückwärts gerichtete Breitensuche.

1. Die Liste aller sicheren Klassifikatoren, deren E_i mit dem Endpunkt F_S matcht sei:
$$c_i = (C_i, A_i, E_i^{ant} \text{ für } i = 1, \cdots, n) \tag{47}$$
2. Berechne alle $S_1^{ant}, \cdots, S_n^{ant}$ mittels $S_i^{ant} = passthrough(F_S, C_i)$
3. Ist der Startpunkt $S_N \in \{S_1^{ant}, \cdots, S_n^{ant}\}$, beende die Suche. Ansonsten führe 1. und 2. für $S_1^{ant}, \cdots, S_n^{ant}$ statt F_S aus, bis S_N gefunden oder eine maximale Anzahl von Suchschritten max^{depth} erreicht ist.

Wurde S_N gefunden, führt das ACS alle Aktionen A, beginnend mit dem letzten aus.
Eine dritte Möglichkeit ist die Kombination beider Algorithmen, die bidirektionale Suche.

5.3 Probleme der Suche

Der Speicherbedarf aller 3 Algorithmen steigt exponentiell, deshalb wird max^{depth} sehr klein gewählt.
Außerdem haben alle 3 Suchverfahren das Problem der Schleifenbildung. Durch die rekursive Suche kommt das Verfahren in Zustände, in denen die Suche bereits war.
Dies kann vermieden werden, indem man die Rekursion nur in den Zuständen fortsetzt, die zuvor noch nicht Ausgangspunkt der weiteren Suche waren.
Auch dies wird wieder über eine Liste realisiert.

6 Wertung von ACS

Der Schwerpunkt der Betrachtungen Stolzmanns liegt eindeutig im Nachempfinden kognitiver Intelligenz. Insbesondere das antizipative Lernverhalten steht im Mittelpunkt der Betrachtung.
ACS sind in der Lage latent zu lernen, sich also ein Modell der Umwelt zu bilden, ohne einen Anreiz in Form von Belohnung oder Bestrafung zu erhalten.
Das Lernen mit Belohnung ist ebenfalls vorgesehen, hat aber nur einen niedrigen Stellenwert, da hierbei die Besonderheit, das Vorwegnehmen bzw. Antizipieren der Auswirkungen eine geringere Rolle spielt.

6.1 Beispiele von Stolzmann

Folgende Beispiele (Tabelle 3) nutzt Stolzmann, um die einzelne Algorithmen eines ACS einzuführen. In diesen Labyrinthen gilt F als Zielzustand.
Stolzmann definiert als Qualität die Menge der gelernten Umgebung, idealerweise 100%. Dabei gilt als gelernt, wenn das System einen sicheren Klassifikator von S_i nach S_j kennt, mit $j \neq j$. Ist dies für alle Übergänge gegeben, gilt die Umgebung als vollständig gelernt.
Ein weiteres Maß ist die Anzahl der Schritte zum Ziel F. Dieses Maß ist aber nur sinnvoll, sofern eine Belohnung im Zielzustand existiert, also $r \neq 0$.
Das erreichte Wissen über die Umwelt hängt außerdem sehr stark von den eingestellten Parametern ab.

Umgebung	Anwendbarkeit
0 0 0 0 0 0 0 6 4 3 2 0 0 0 5 0 F 0 0 0 0 0 0 0	Stolzmann zeigt, dass ein einfaches ACS in der Lage ist 100% der Umwelt zu lernen, sofern keine Belohnung gegeben wird. Wird eine Belohnung gegeben, dominiert sie die Qualität recht schnell, so dass das ACS die Umwelt nicht mehr erforscht, sondern direkt zum Ziel steuert.
0 0 0 0 0 0 0 7 5 4 3 0 0 0 6 0 2 0 0 0 0 0 F 0 0 0 0 0 0 0	In diesem Labyrinth erreicht das ACS kein hundertprozentiges Wissen. Der Klassifikator $\begin{matrix}0&0&\#\\1&1&\#\\0&1&\#\end{matrix} \Downarrow \begin{matrix}1&1&\#\\0&1&\#\\0&F&\#\end{matrix}$ kann in 3 und 5 angewandt werden, ist aber nur korrekt im Zustand 3. Mit dem Verfahren Spezifikation von konstanten Elementen kann ein ACS auch dieses Problem lösen und erreicht 100% Lernerfolg.
0 0 0 0 0 0 0 9 6 5 4 0 0 0 7 0 3 0 0 0 8 0 2 0 0 0 0 0 F 0 0 0 0 0 0 0	Ein Klassifikator wie $\begin{matrix}0&0&\#\\1&1&\#\\\#&\#&\#\end{matrix} \Downarrow \begin{matrix}1&1&\#\\0&1&\#\\\#&\#&\#\end{matrix}$ trifft in 4,5 und 6 zu, ist aber nur in 4 korrekt. Wird er in 5 angewandt, erhält der Klassifikator eine Marke. Wird er später im Zustand 6 angewandt, kann nur mittels generalisierten Marken auch dieser Zustand zusätzlich gespeichert werden. In diesem Labyrinth profitiert ein ACS also von dieser Methode.
0 0 0 0 0 0 0 11 7 6 5 0 0 0 8 0 4 0 0 0 9 0 3 0 0 0 10 0 2 0 0 0 0 0 F 0 0 0 0 0 0 0	Die Zustände 3 und 9 sind für das System nicht zu unterscheiden. Deshalb schlägt die Spezifikation von konstanten Elementen fehl. Hier kann das System nur mittels Aktionsketten lernen, ansonsten lernt es Endlosschleifen.

Tabelle 3: Beispielszenarien und Anwendbarkeit von ACS

6.2 Weitere Beispielszenarien

Die bisherigen Beispiele sind für ACS sehr gut geeignet. In ihnen kann ein ACS die Umwelt gut lernen. Es gibt aber auch Szenarien, in denen ein ACS nicht in der Lage ist, sich ein Modell der Umgebung zu bilden (Tabelle 4)

0 0 0 0 0 0 1 S_1 S_2 S_3 S_4 S_5 0 0 0 0 0 0	Lange Gänge bestehen nur aus Alias-Zuständen. Egal, welche Aktion ausgeführt wird, der Zustand bleibt konstant. Sämtliche Klassifikatoren würden durch die Mechanismen des *useless cases* geschwächt (Gleichung 12). Es ist ersichtlich, dass ein ACS nicht gut für diese Art von Umgebung geeignet ist.
0 0 S_N 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 S_0 0 0 S_1 0 0 S_2 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 F_S 0	Zusätzlich zu den eben genannten Problemen in langen Gängen haben Alias-Zustände eine weitere negative Eigenschaft. Auch die Handlungsplanung schlägt fehl. Die Alias-Zustände werden nicht übersprungen, da das ACS keine guten Klassifikatoren lernen konnte. Deshalb würde eine Suche, die bereits in S_0 oder S_1 war, bei Erreichen von S_2 abbrechen, um Schleifen zu vermeiden.

Tabelle 4: problematische Beispielszenarien in ACS

Literatur

[CR94] D. Cliff and S. Ross. Adding memory to zcs. Adaptive Behavior, 3(2):101–150., 1994.

[DB94] M. Dorigo and H. Bersini. A comparison of q-learning and classifier systems. In From Animals to Animats, Third International Conference on Simulation of Adaptive Behaviou, August 1994. 69, 1994.

[Hof93] J. Hoffmann. Vorhersage und erkenntnis [anticipation and cognition]. Goettingen, Germany: Hogrefe., 1993.

[Hol80] J. Holland. Adaptive algorithms for discovering and using general patterns in growing knowledge bases. International Journal for Policy Analysis and Information Systems, Vol 4, No 3, 1980. pp 245–268., 1980.

[KL99] Tim Kovacs and Pier Luca Lanzi. A Learning Classifier Systems Bibliography. Technical Report CSRP-99-19, School of Computer Science, University of Birmingham, 1999.

[lcs] learning classifier systems.org. online im Internet, http://www.learning-classifier-systems.org/.

[pre] Erkenntnisse aus der Psychologie fließen in die Robotik ein. online im Internet, http://www.psychologie.uni-wuerzburg.de/stolzmann/pressetext.htm.

[SBHG00] Wolfgang Stolzmann, Martin Butz, J. Hoffmann, and D. E. Goldberg. First cognitive capabilities in the anticipatory classifier system. In *IlliGAL report 2000008*, pages 287–296. University of Illinois at UrbanaChampaign: Illinois Genetic Algorithms Laboratory, 2000.

[Sew49] J. Seward. An experimental analysis of latent learning. Journal of Experimental Psychology 39, pp. 177-186., 1949.

[Sto] Wolfgang Stolzmann. An introduction to anticipatory classifier systems. online im Internet, http://link.springer.de/link/service/series/0558/papers/1813/18130175.pdf.

[Tol32] Edward C. Tolman. Purposive behaviour in animals and men, 1932.

A Definitionen

c_i	:	der i-te Klassifikator
C_i	:	Bedingung von c_i
A_i	:	Aktionen von c_i
E_i	:	Erwartung von c_i
δ^{delete}	:	Schwellwert für einen unzuverlässigen Klassifikator
ρ_t	:	Belohnung der Umwelt zum Zeitpunkt t
σ_t	:	Zustand der Umwelt zum Zeitpunkt t
S_t	:	Messwerte der Sensoren zum Zeitpunkt t
α	:	Wirkungen die die Aktoren auf die Umwelt ausüben
α_{max}	:	Maximale Anzahl von Aktion einer Aktionskette
q	:	Qualität des Klassifikators
r	:	erwartete Belohnung des Klassifikators
max^{depth}	:	maximale Suchtiefe bei der Handlungsplanung
ϵ^{sure}	:	maximale Unsicherheit eines sicheren Klassifikators
M	:	Message-Liste zur Kontrolle von Aktionsketten
f_i	:	Fitness eines Klassifikators